AD720316

FINAL REPORT - TASK AREA I
(Volume IV)

(21 June 1968 - 31 December 1970)

FOR THE PROJECT
RESEARCH IN MACHINE-INDEPENDENT
SOFTWARE PROGRAMMING

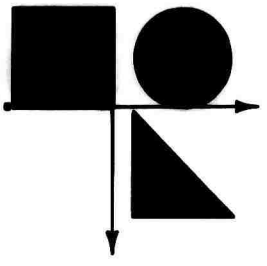
DDC
RECEIVED
MAR 19 1971
D

Massachusetts

COMPUTER ASSOCIATES

division of

APPLIED DATA RESEARCH, INC.



Reproduced by
NATIONAL TECHNICAL
INFORMATION SERVICE
Springfield, Va. 22151

71

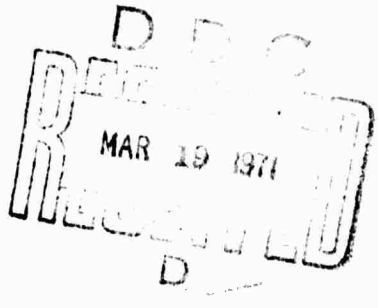
APPLIED DATA RESEARCH, INC.

LAKESIDE OFFICE PARK, WAKEFIELD, MASSACHUSETTS 01880 · (617) 245-9540

FINAL REPORT - TASK AREA I
(Volume IV)

(21 June 1968 - 31 December 1970)

FOR THE PROJECT
RESEARCH IN MACHINE-INDEPENDENT
SOFTWARE PROGRAMMING



Principal Investigators:

Task Area I	Carlos Christensen	(617) 245-9540
Task Area II	Anatol W. Holt	(617) 245-9540

Project Manager:
Robert E. Millstein (617) 245-9540

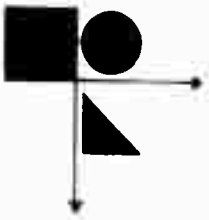
ARPA Order Number - ARPA 1228
Program Code Number - 8D30

Contractor:	Massachusetts Computer Associates, Inc., Division of ADR
Contract No.:	DAHC04-68-C-0043
Effective Date:	21 June 1968
Expiration Date:	30 September 1971
Amount:	$696,800.00

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

Sponsored by
Advanced Research Projects Agency
ARPA Order Number - 1228

CA-7102-2614



APPLIED DATA RESEARCH, INC.

LAKESIDE OFFICE PARK, WAKEFIELD, MASSACHUSETTS 01880 · (617) 245-9540

A REPORT ON AMBIT/G
(Volume IV)

by

Carlos Christensen

Michael S. Wolfberg

Michael J. Fischer*

CA-7102-2614
February 26, 1971

* Consultant to Applied Data Research, Inc.
Address: Department of Mathematics, M.I.T.,
Cambridge, Massachusetts

This is the fourth of four volumes of the final report on Task Area I of the project "Research in Machine-Independent Software Programming". This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by U.S. Army Research Office-Durham, Box CM, Duke Station, Durham, North Carolina 27706, under Contract DAHC04-68-C-0043.

CONTENTS

Volume I

Abstract

1. Summary 1

2. Fundamentals 3
data graph, constraints, program, general philosophy specific languages.

3. Representation of Programs 15
overview, program syntax, correspondence between program graphs and diagrams.

4. The Interpreter 30
overview, the compiler, interpretation of 'linkrep's, user-defined functions, error messages.

5. The Loader 48
overview, error messages, loader syntax, sample encodement, sample error.

6. Initialization and the Built-in System 65
hints, built-in nodes, built-in links, built-in function definitions, built-in rules, built-in data, built-in functions, sample error.

7. The Debugging Facility 99
lexical conventions, statements, statement forms, sample session.

8. The Implementation 114
credits and acknowledgements, internal view, files, PL/I data formats, PL/I implementation of the interpreter and loader.

9. Further Work 152

10. Project Bibliography 157

Volume II

11. Examples of AMBIT/G Programs
observations, introductory examples: reversing a list, two forms of input, function calling, LISP garbage collector, another garbage collector, an interactive program, sorting, factorial computation and recursion.

Volume III

12. The AMBIT/G Interpreter as an AMBIT/G Program
description, listing.

Volume IV

13. The AMBIT/G Loader as an AMBIT/G Program
description, listing.

CHAPTER 13

THE AMBIT/G LOADER AS AN AMBIT/G PROGRAM

This final volume of "A Report on AMBIT/G" describes the AMBIT/G loader as an AMBIT/G program, which is, in fact, how it is implemented. We begin by giving a verbal description of the loader. Then each function of the loader is briefly discussed. The remainder of the volume consists of many pages of diagrams which constitute the listing of the loader.

The reader who wants to understand the program (rather than just look at pretty pictures) is urged to be familiar with the contents of Chapter 5 of this report in Volume I. Although that chapter attempted to describe completely the syntax of loader input, including lexical conventions, it is the listing of the loader which can be used as a final reference for obscure issues.

A table of contents for the listing is given at the beginning of the listing.

A DESCRIPTION

The AMBIT/G loader consists of the function 'load' and nine other major functions. In addition, the loader contains calls on the 'eq_any' function which determines whether its first argument is the same node as any of its other arguments; its result is 'boolean true' if it is the same and 'boolean false' otherwise. The listing of the loader does not include the 'eq_any' function definition. The functions 'tail_1', 'head_1', and 'head_2' are used to transmit arguments and results, and these functions are also omitted from the listing. One additional kind of omission in the listing is the write calls on built-ins 'read_function' and 'write_function' to define the functions of the loader and the links which it uses.

Execution of the loader begins by a trivial initialization and a call on 'get_statement'. The first statement must begin with a minus sign and must be a '-page-' statement or a '-start-' statement. If a '-start-' statement is found, the remainder of the statement is used to locate a 'rule' node to be returned as the result of the 'load' function, and control returns to the caller. Otherwise, the processing of a page begins.

The loader looks for a next node by calling 'get_statement', and if that statement begins with a minus sign it checks for the following possibilities: '-rule-', '-ruleref-', '-links-', '-start-', or '-page-' . Finding either '-rule-' or '-ruleref-' causes the loader to treat the specified rule or rule reference as a glorified node. The '-links-' statement causes the loader to start to look for a next link specification for the current loader page. A '-start-' or '-page-' statement causes the loader to finish up processing of the current page, and then that statement is appropriately interpreted.

When seeking a next node, if a statement does not begin with a minus sign it is taken as a specification of a data node on the loader page. It must have a page-name and type, and optionally a subname.

During the reading of nodes of a loader page, a 'pname' list is developed as a symbol table for the duration of the page. The list consists of nodes of type 'pname' connected by 'next' links. The list is logically terminated by a 'pname' link of a 'pname' node pointing to 'charconn end' which represents

a null page-name. Each entry of this list is represented by a 'pname' link pointing to a list of 'char' nodes connected by 'charconn's and a 'node' link off of the same 'pname' node pointing to the very node represented by the 'pname'.

The physical 'pname' list grows by locating another 'pname' node at the end of the list when necessary, but it never shrinks. The physical end is represented by a 'pname' node's 'next' link pointing to 'pname end'. At the end of processing a loader page the logical 'pname' list is made empty, but the physical 'pname' list is not affected.

When a '-rule-' statement is encountered, the logical 'pname' list grows by one entry for the 'rule' node but is then frozen for the duration of the rule specification until an '-endrule-' statement is detected. The page-names used within a rule are local to the rule and are saved by the loader on the physical 'pname' list immediately following the frozen list. The 'node' link of each 'pname' node used to represent a node in a rule points at a 'noderep' node since each node mentioned in a rule is represented by a 'noderep' node. The 'rep' link of such a 'noderep' representing a fully named node points to that node, and the 'variability' link of that 'noderep' points to 'flag fixed'. If a rule contains a particular fully named node in more than one place, different entries are used on the 'pname' list, but they all share one 'noderep' node. This merging of split fully named nodes also is done for link names and for any implicitly mentioned nodes, such as those required in an expansion of a node in a rule with a type but no subname. The 'pname' link of 'pname' nodes representing these implicitly specified fully named nodes points to 'charconn free' as a special indicator.

During the processing of a rule, there are six types of statements which may be encountered while seeking a next node:

a) fully named node
b) node with type, but no subname
c) node with type to be tested, but no subname
d) unnamed node
e) ' --- ' statement
f) '-endrule-' statement

Any node specification causes an entry to be placed on the 'pname' list. In case b or c an extra link is pushed down on the 'contents' list of links hanging off the 'rule' node which represents the rule being loaded. In either case, its name is 'link type' and its destination is the appropriate 'type' node. For case b its mode is 'frame', and for case c its mode is 'test'. At this point, no other links are represented on the 'contents' list.

Unnamed nodes cannot be split in a rule since there is no syntax for this.

The ' --- ' statement signals the end of node specifications and causes the loader to expect link specifications for the rule.

If an '-endrule-' statement is encountered, the rule contains no links; the temporary end of the 'pname' list is eliminated, and the loader proceeds to seek a next node on the loader page.

During the processing of links of a rule, each statement specifies one link whose representation is constructed and pushed down on the 'contents' list of links hanging off the 'rule' node which represents the rule being loaded. A link which is specified by a textual name (rather than an explicit spur to a node) constitutes an implicit use of a fully named node and thus may introduce another entry on the 'pname' list. The only other sort of statement expected during the processing of links of a rule is '-endrule-', whose effects were already described.

When a '-ruleref-' is encountered by the loader when it is seeking a next node, an entry is added to the 'pname' list for the referenced rule, but no links of the 'rule' node representing the rule are modified.

When the loader looks for a next link on a loader page, it calls 'get_statement', and if that statement begins with a minus sign it checks for the following possibilities: '-deflinks-', '-start-', or '-page-'. A '-deflinks-' statement causes a write call on the built-in 'read_function' and a write call on the built-in 'write_function' for each link to be defined. If the statement does not begin with a minus sign it specifies the setting of a link. This is

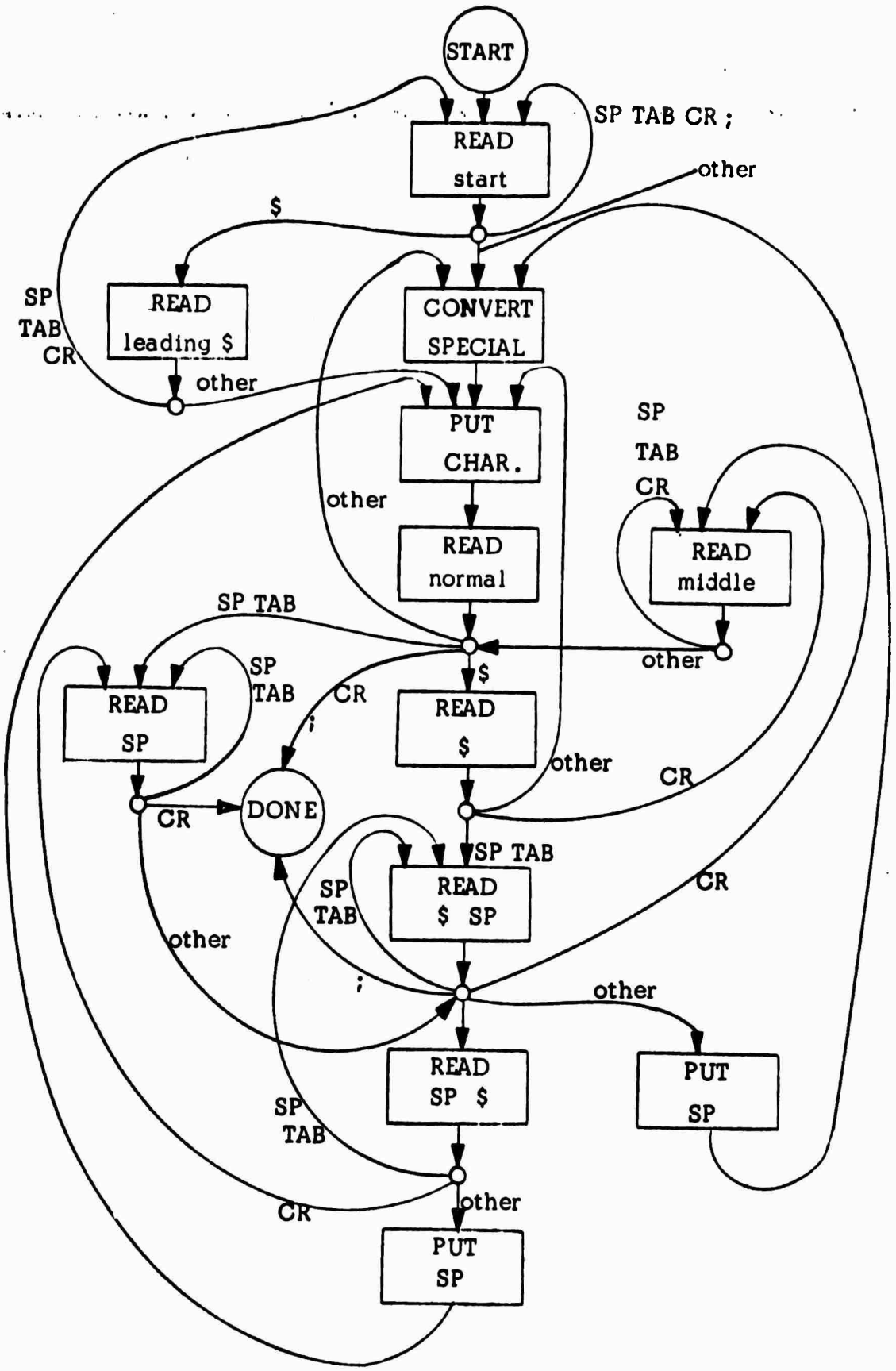
done by the loader's performing a write call on the built-in 'link''.

Throughout the operation of the loader, 'charconn' nodes are constantly being taken from and placed back on the free list of 'charconn' nodes. This list was initially created by the initializer with a reasonable subset (100) of the 'charconn's. When the loader seeks a next 'charconn' from the free list it calls the 'next_charconn' function which may have to locate another 'charconn' if the list is empty. After processing many types of statements, after processing a rule, and after processing a page 'charconn' nodes are returned to the free list.

This description of the loader as a program will conclude with individual descriptions of the nine major functions of the loader.

get_statement

This function is responsible for reading one character at a time from the input source file and gathering together a loader statement as the result of the function. It tallies each statement in the counter 'circle line'. It ignores (but tallies) comment statements. It converts unprotected special characters into nodes of type 'special'. It is concerned mostly with spaces, tabs, dollar signs, semicolons, new-lines, and special characters. Leading and trailing spaces and tabs are ignored. Spaces and tabs are not distinguished, and any amount of space in the middle of a statement is converted to one space (the node 'special SP'). Protective dollar signs are removed, and line continuations are processed. The following state diagram is the best description of this function; the AMBIT/G code closely corresponds to the diagram.


START
READ
start
SP TAB CR ;
other
$
SP
TAB
CR
READ
leading $
other
CONVERT
SPECIAL
PUT
CHAR.
other
READ
normal
SP
TAB
CR
READ
middle
SP TAB
other
READ
SP
SP
TAB
CR
;
READ
$
$
other
CR
CR
DONE
SP TAB
SP
TAB
READ
$ SP
CR
other
;
other
READ
SP $
PUT
SP
SP
TAB
CR
other
PUT
SP

get_word

This function has no argument, but it assumes that 'circle char' points to a partial statement. Its first result is a pointer to the 'charconn' node pointed to by 'circle char'. Its second result is the last 'charconn' node in the list of 'charconn's whose 'value' link points to a 'char' node. Thus the two results point to the beginning and end of the next word in the statement. The reason for the scan to stop is either of the following:

a) 'charconn end' is encountered which indicates the end of the statement; then 'circle special' is made to point at 'charconn end'.
b) a 'charconn' node is encountered whose 'value' link points to a 'special' node; then 'circle special' is made to point to that 'special' node.

In either case, the 'next' link of the 'charconn' node which is the second result is made to point to 'charconn end'.

This function considers an error condition to hold if either 'circle char' points to an empty list or if the first element of the list is not a 'char' node.

find_node

This function has no argument, but it calls on 'get_word' to read the next word of the current statement. It then looks down the 'pname' list for a match of a page-name with the word. If it does find a match, it returns as its result the node represented by the given page-name ; this is found at the destination of the 'node' link of the 'pname' node whose 'pname' link points to the given page-name. If a match is not found, an error condition is reported.

define_node

This function has one argument which is a 'type' node. That is used along with the word at 'circle char' to locate a node which is then pointed to by 'circle node'. Then an entry is added to the 'pname' list with a page-name as the word pointed to by 'circle pname' and its definition as the node pointed to by 'circle node'.

find_noderep

This function returns either an existing 'noderep' in the 'pname' list or a freshly inserted one which is a node representation of the given argument. Thus this function sometimes serves to define a fully named node of a rule.

get_link_name

This function scans the current statement (using 'circle char') for a link name specified either explicitly as a spur or implicitly as a text string. It has no argument, but its result is the node which names the link.

heads_or_tails

This function processes both the heads and the tails of a link in a rule. It has no argument, but scans the current statement using 'circle char'. Its result is a list of 'diamond' nodes where each 'diamond''s 'value' link points to a 'noderep'.

free_charconn_list

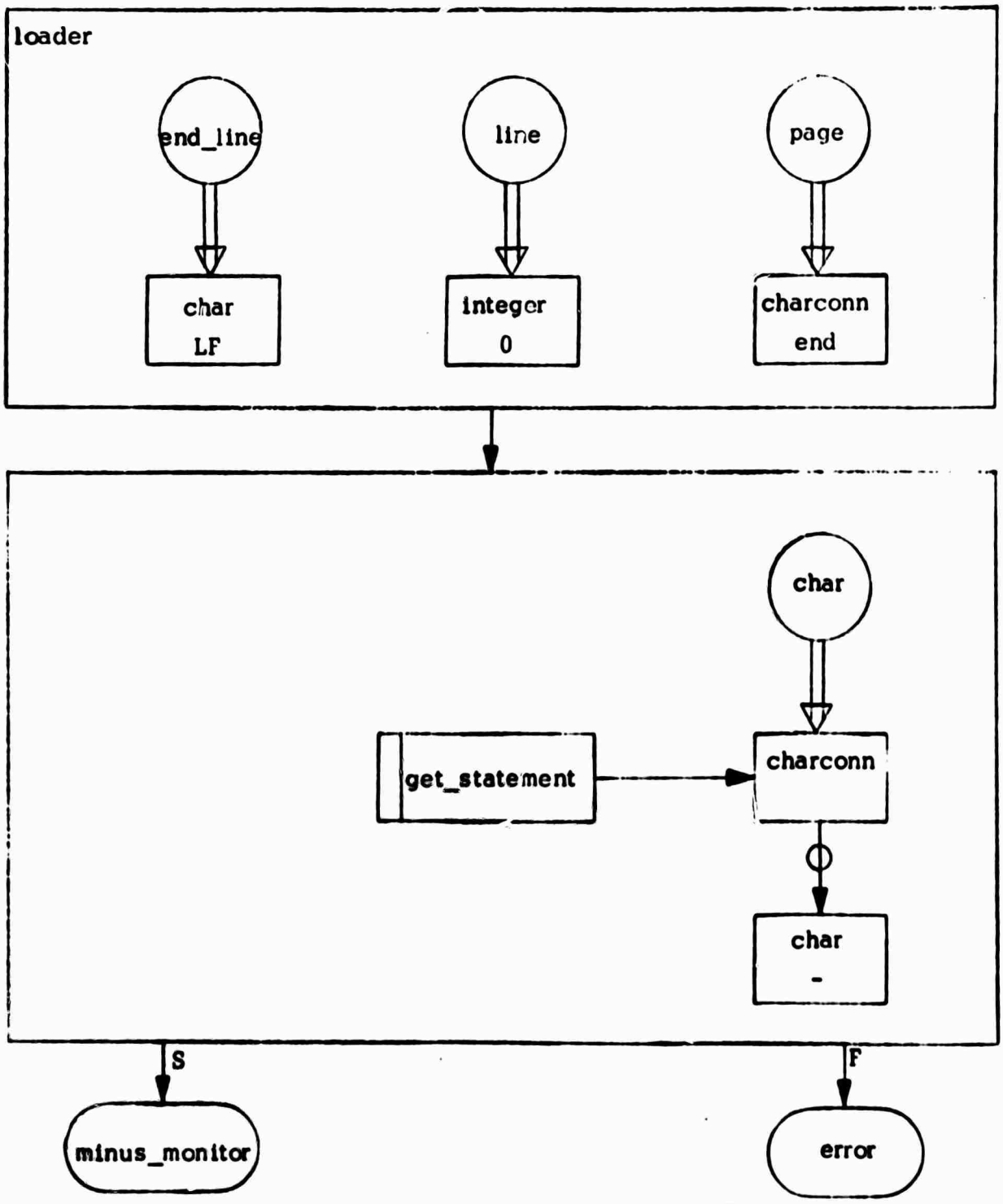
This function returns the list of 'charconn' nodes given as its argument to the free list of 'charconn's. The function has no result.

This function takes an argument which is a node of type 'charconn'. It returns a next 'charconn' node connected to the argument by its 'next' link. Either that next node was already there, or the function locates such a node.

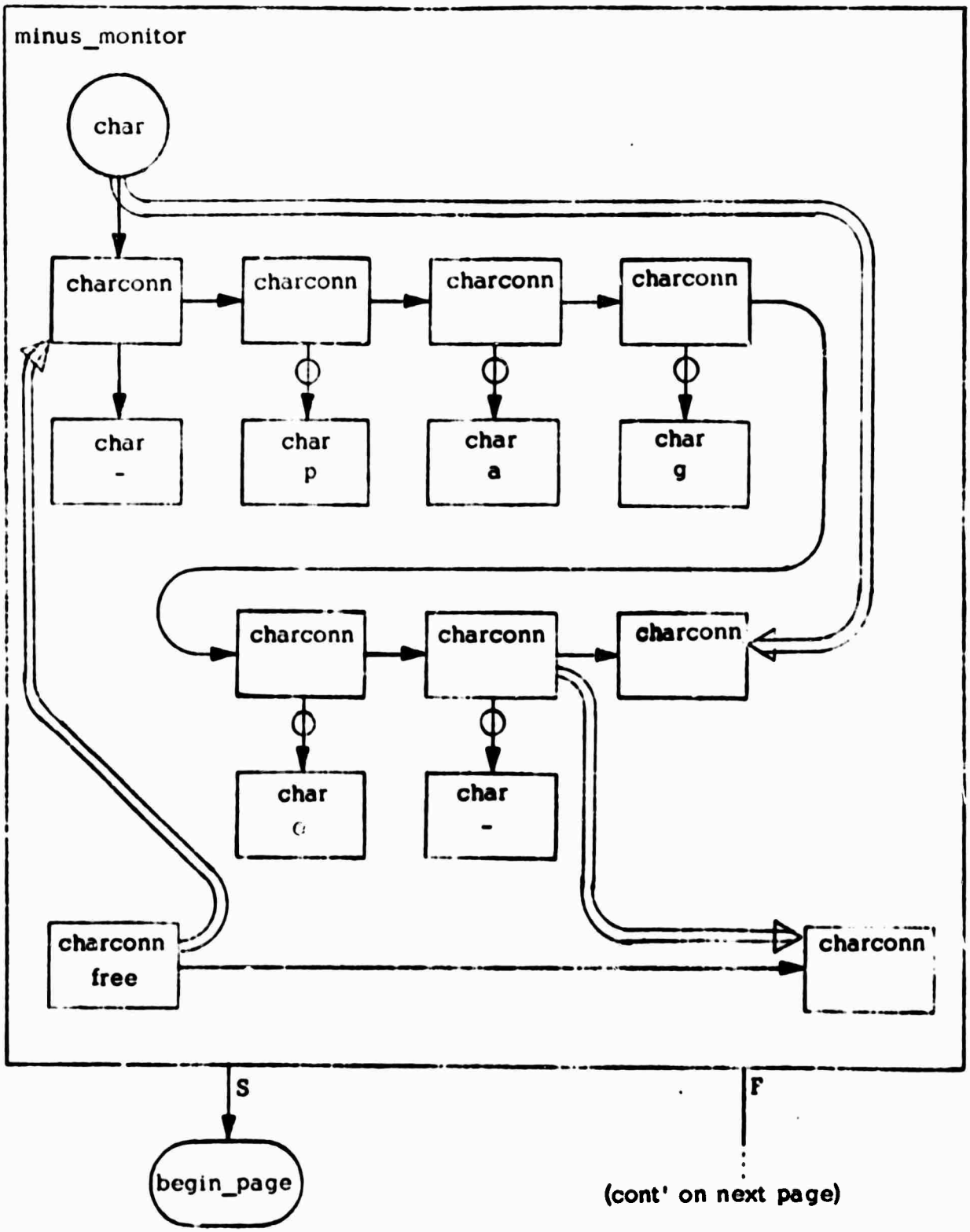
THE LISTING

The remainder of this volume consists of the listing of the AMBIT/G loader. The following is a table of contents for the listing.

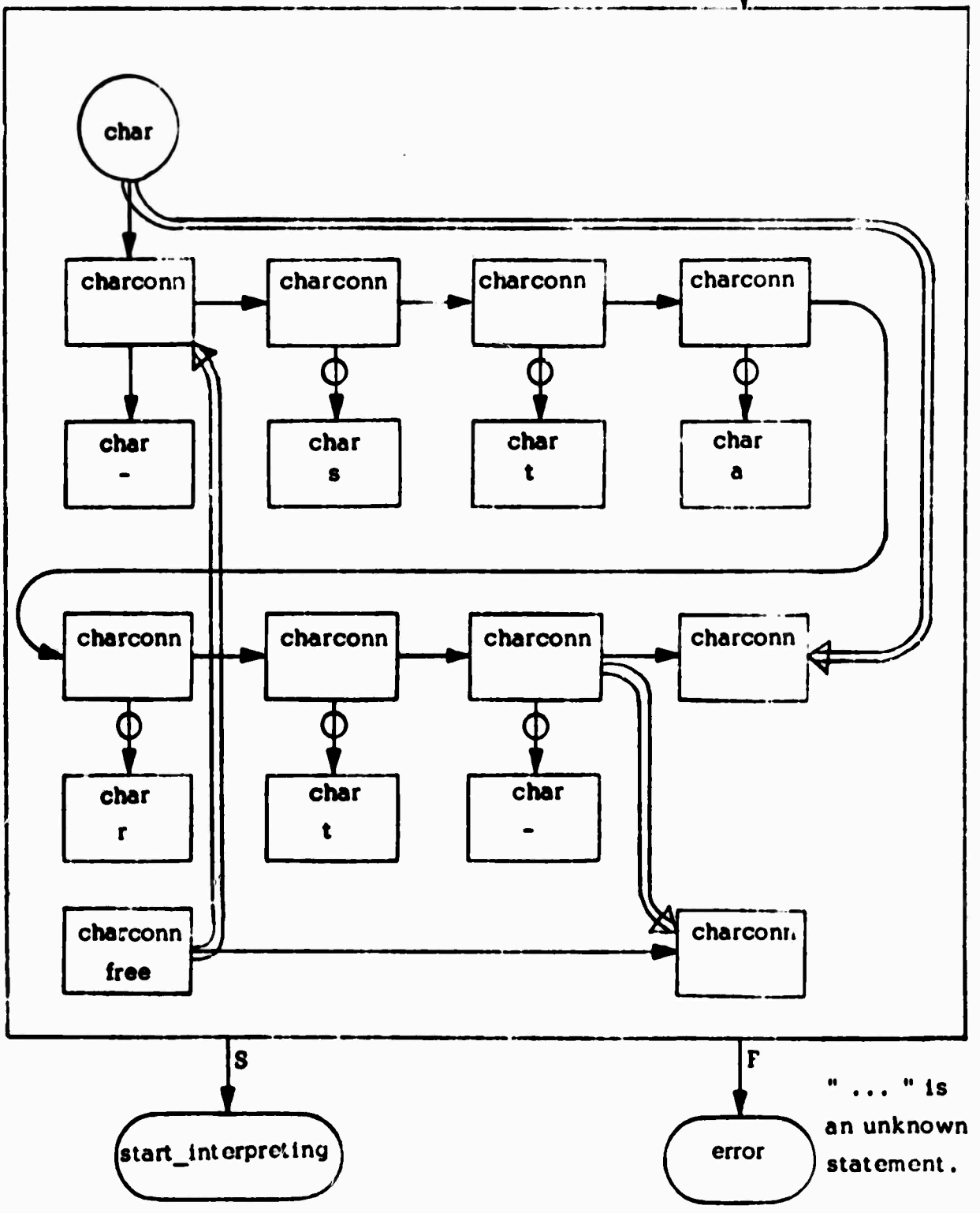
Function	Loader Page Titles	Starting Page Number
loader	L1 - L27	10
get_statement	GS1 - GS8	41
get_word	GW1 - GW2	51
find_node	FN1 - FN3	53
define_node	DN1 - DN2	56
find_noderep	FNR1 - FNR3	58
get_link_name	GLN1 - GLN2	61
heads_or_tails	HT1 - HT3	63
free_charconn_list	FCL1	66
next_charconn	NC1	67

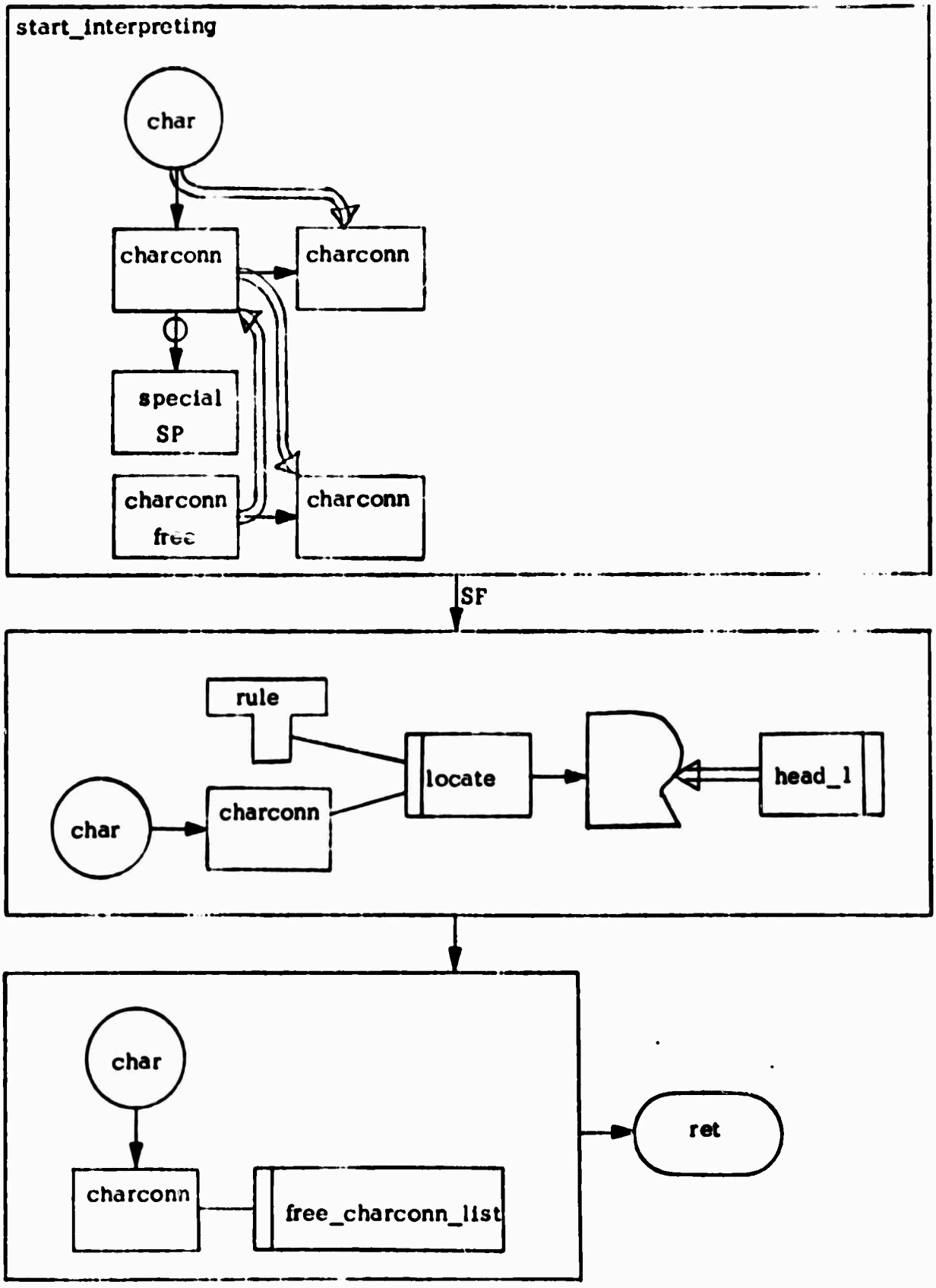


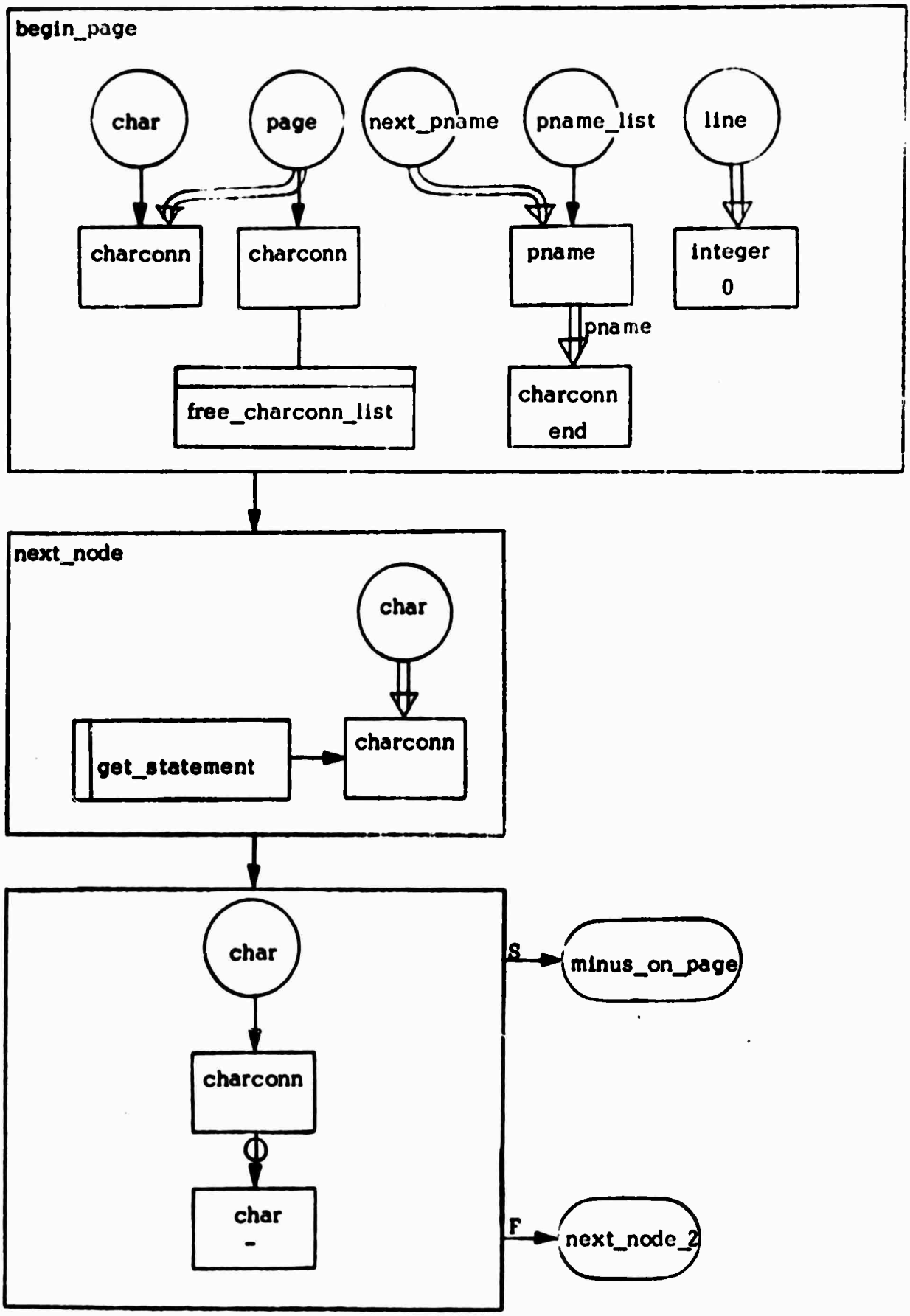
The first statement read by the loader does not begin with a "-".

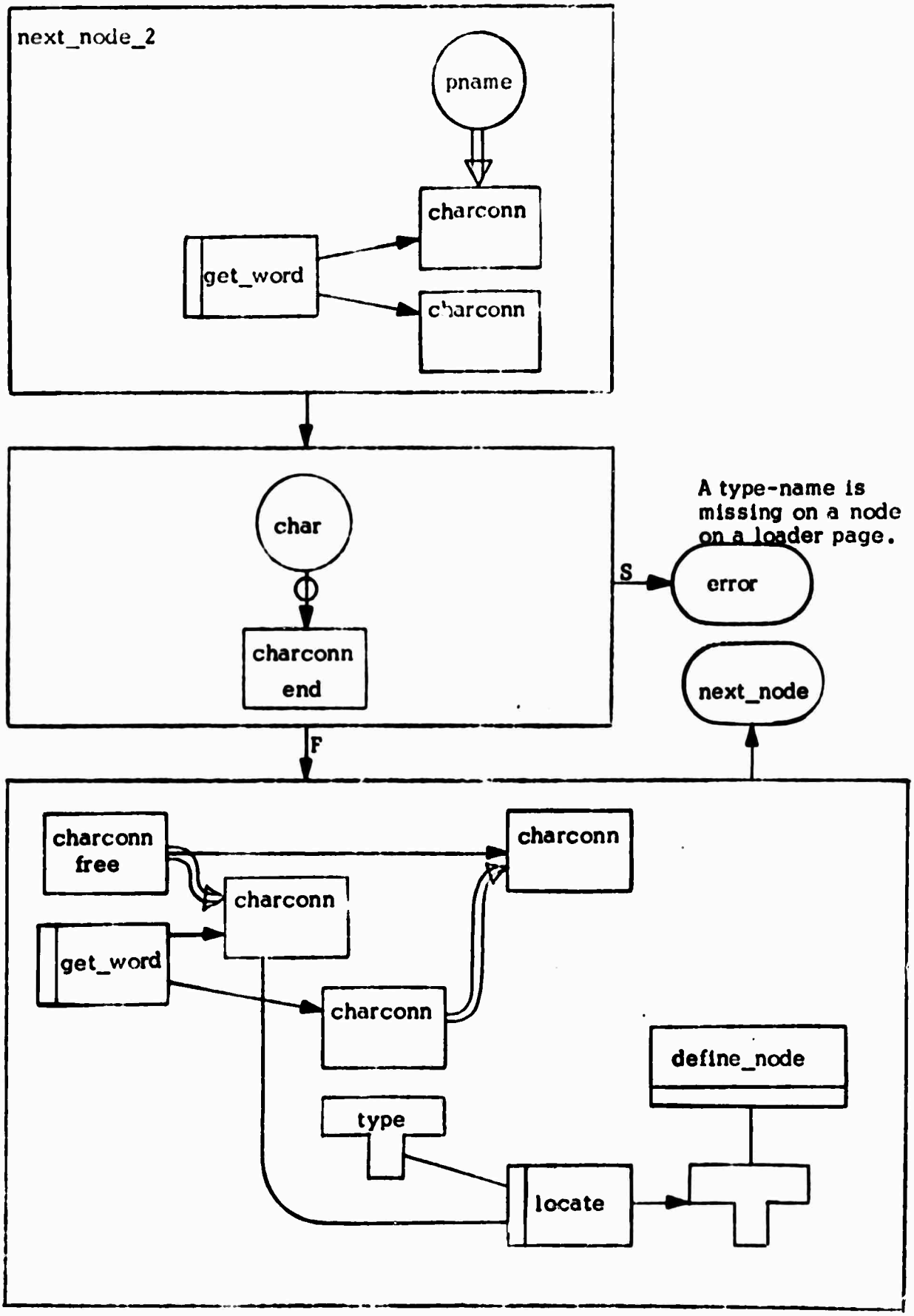

minus_monitor
char
charconn
charconn
charconn
charconn
char
-
char
p
char
a
char
g
charconn
charconn
charconn
char
c
char
-
charconn
free
charconn
S
F
begin_page
(cont' on next page)

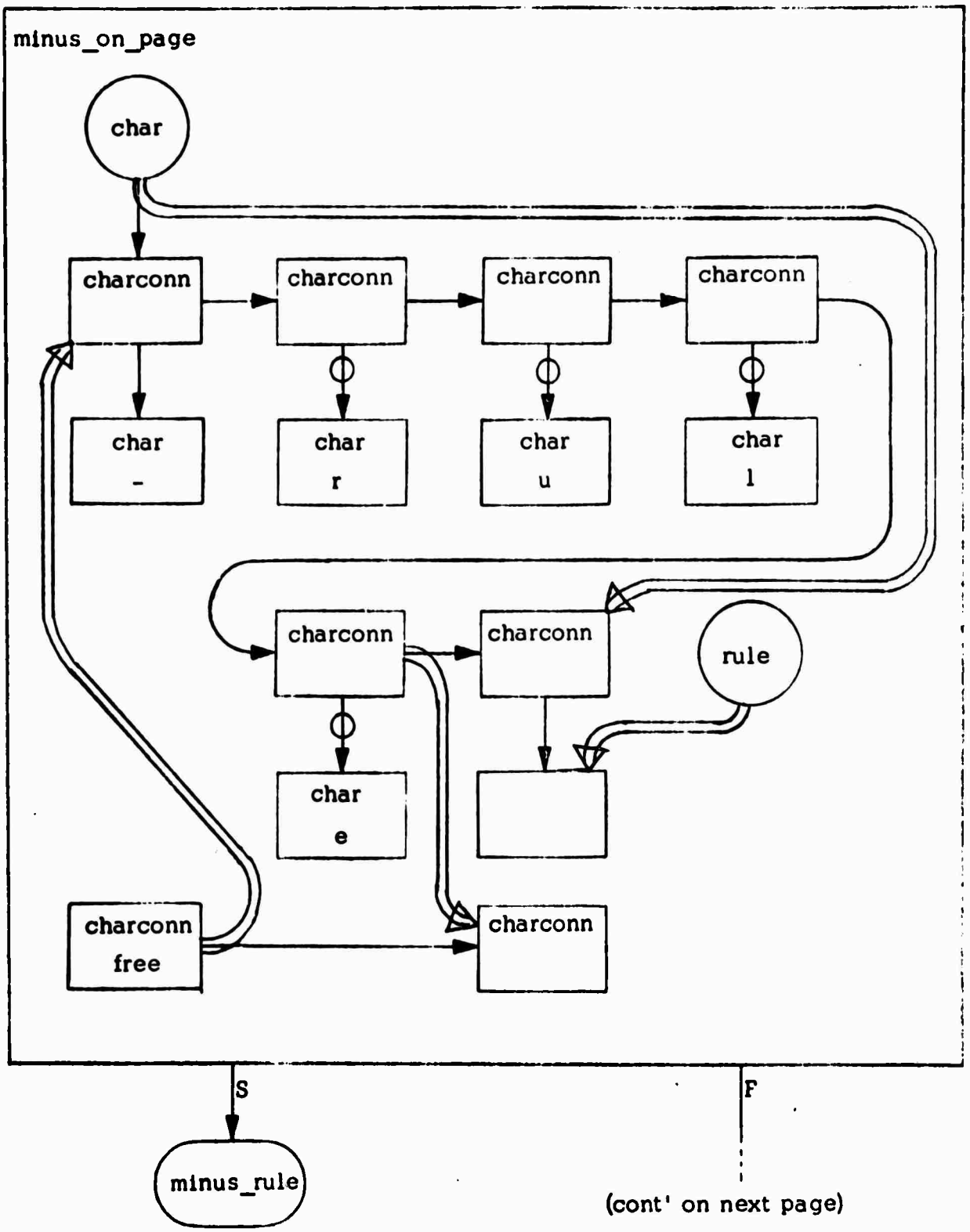
(cont' from previous page)



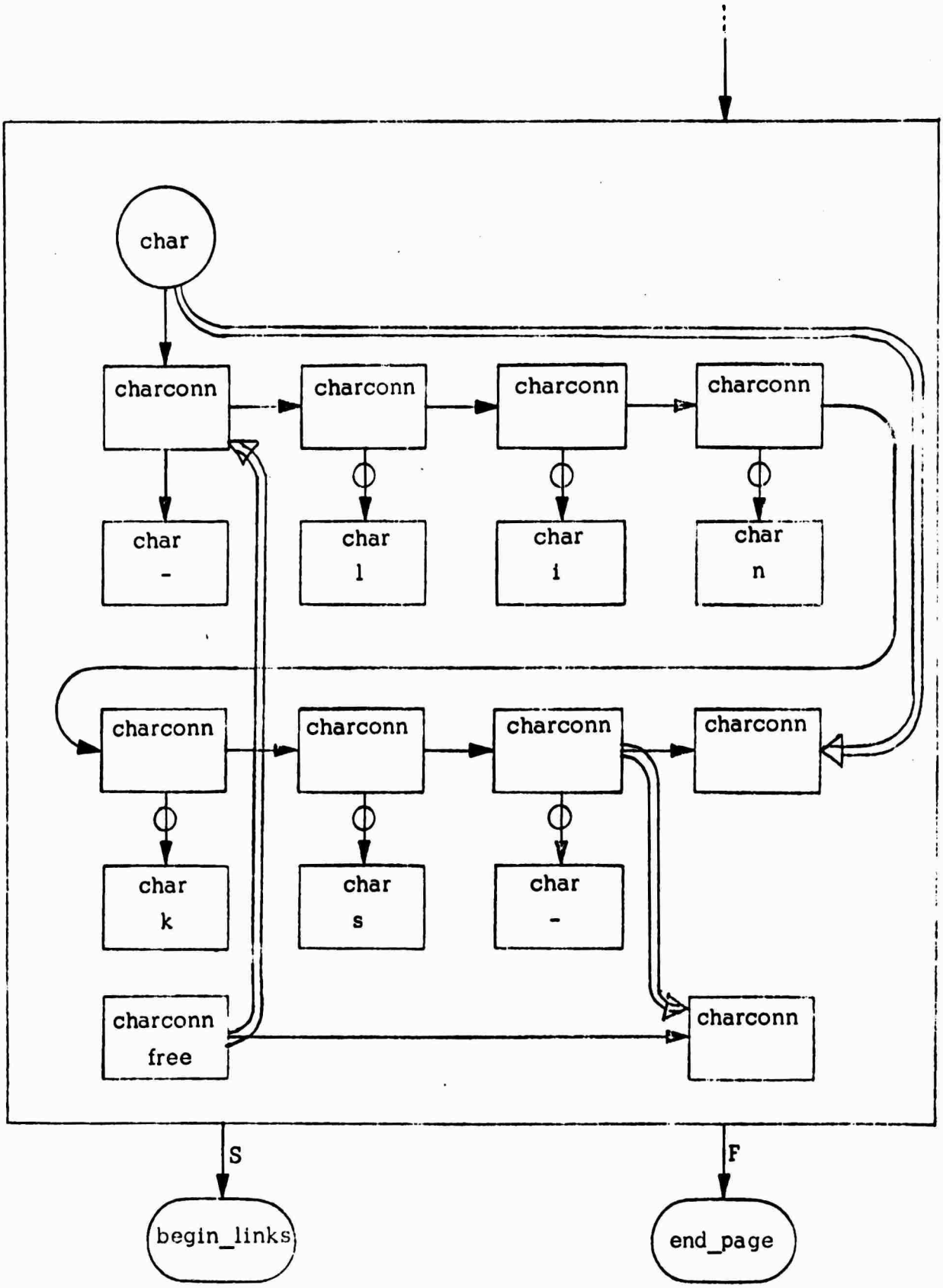

start_interpreting
char
charconn
charconn
special
SP
charconn
free
charconn
SF
rule
char
charconn
locate
head_1
char
charconn
free_charconn_list
ret

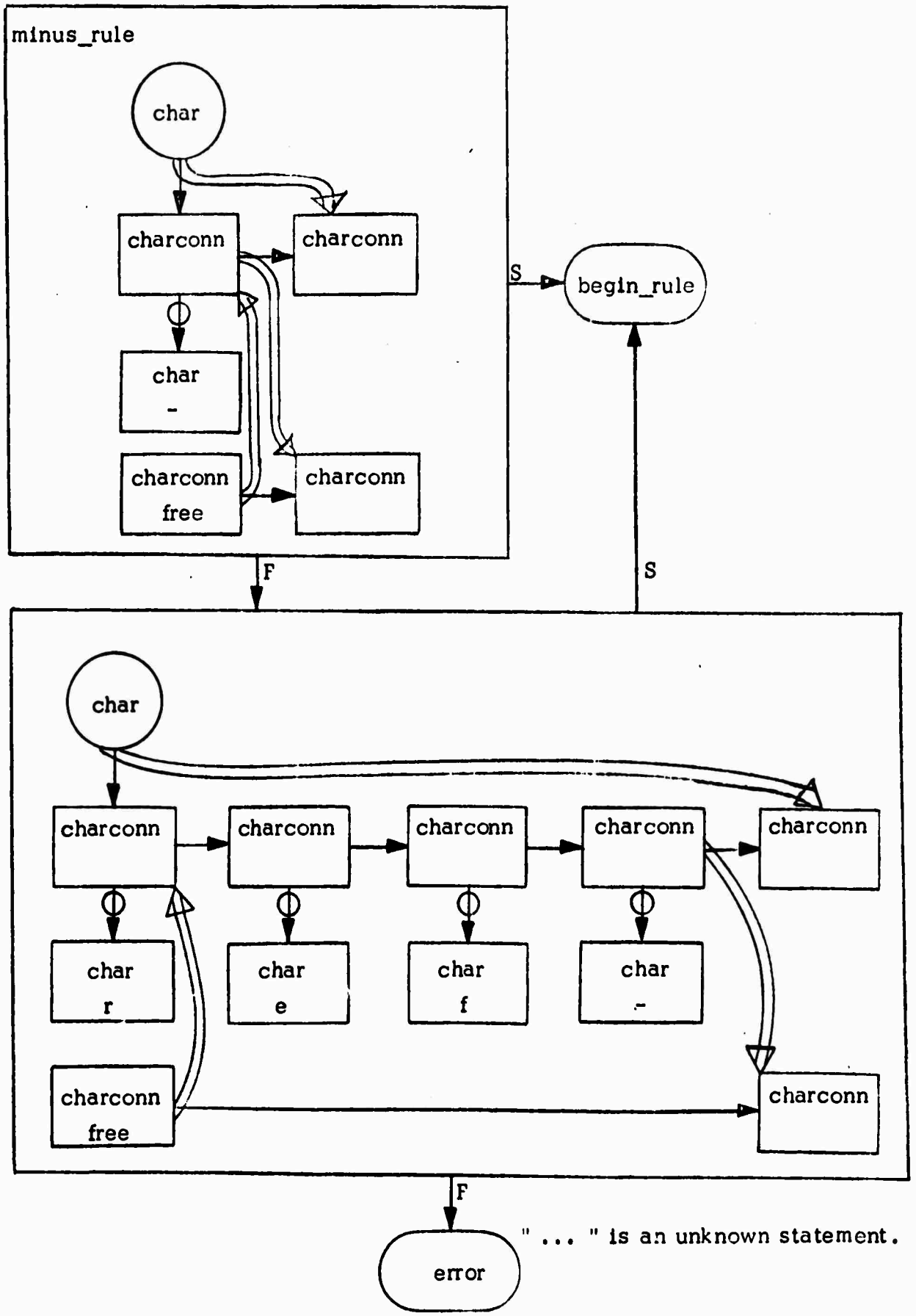

begin_page
char
page
next_pname
pname_list
line
charconn
charconn
pname
integer
0
pname
free_charconn_list
charconn
end
next_node
char
get_statement
charconn
char
S
minus_on_page
charconn
char
-
F
next_node_2

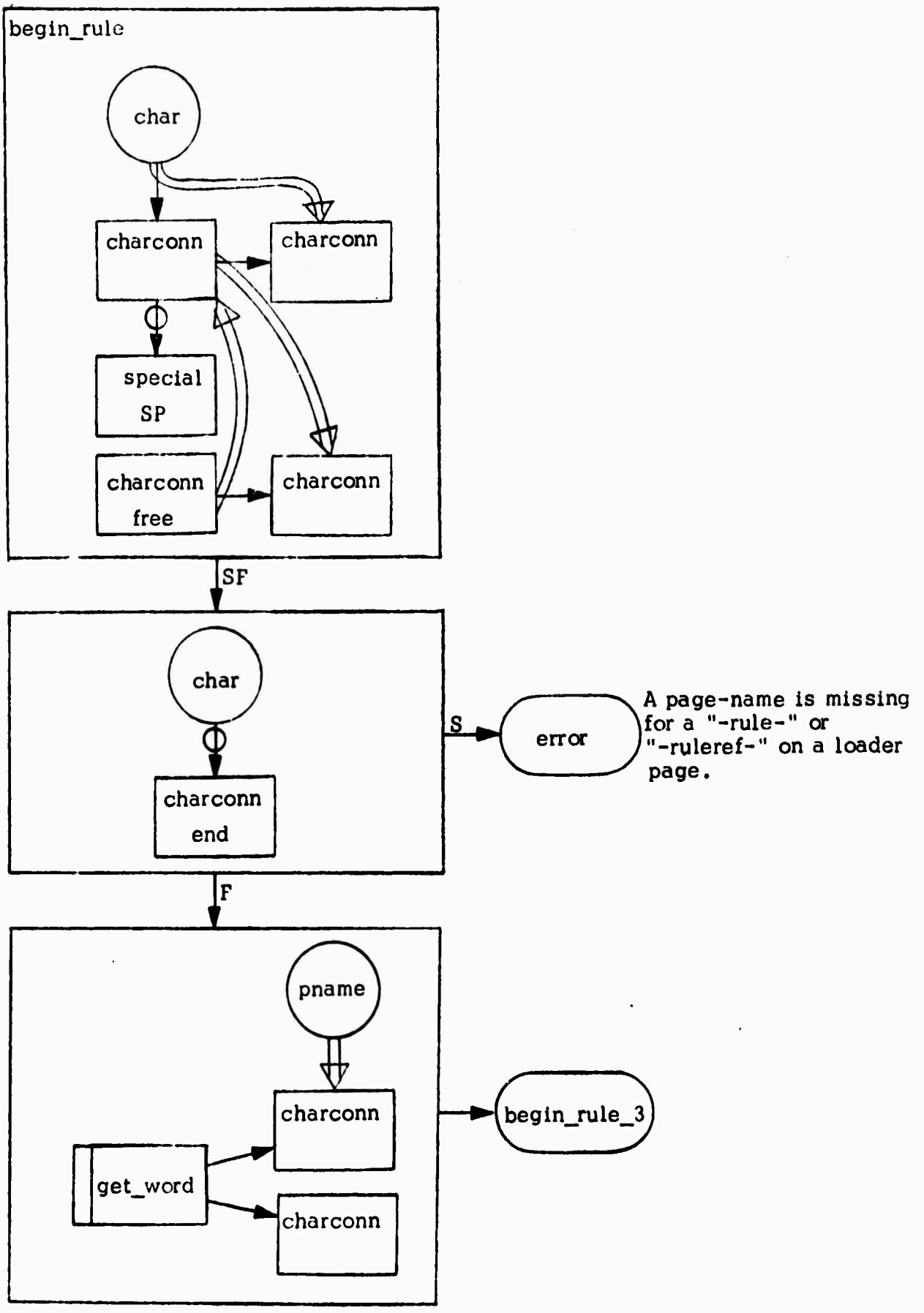

next_node_2
pname
charconn
get_word
charconn
char
charconn
end
A type-name is
missing on a node
on a loader page.
S
error
next_node
F
charconn
free
charconn
charconn
get_word
charconn
define_node
type
locate

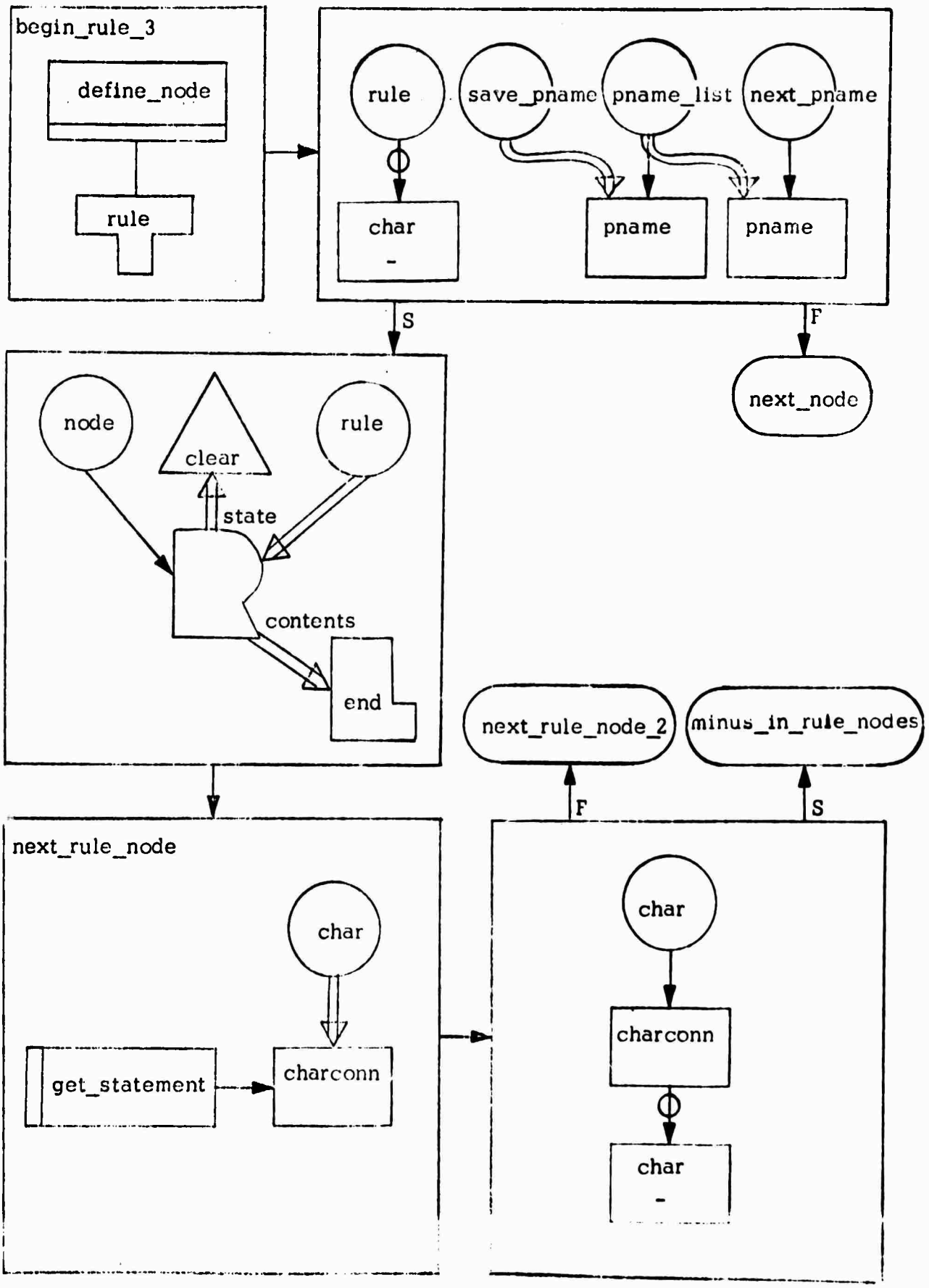

minus_on_page
char
charconn
charconn
charconn
charconn
char
-
char
r
char
u
char
l
charconn
charconn
rule
char
e
charconn
free
charconn
S
F
minus_rule
(cont' on next page)

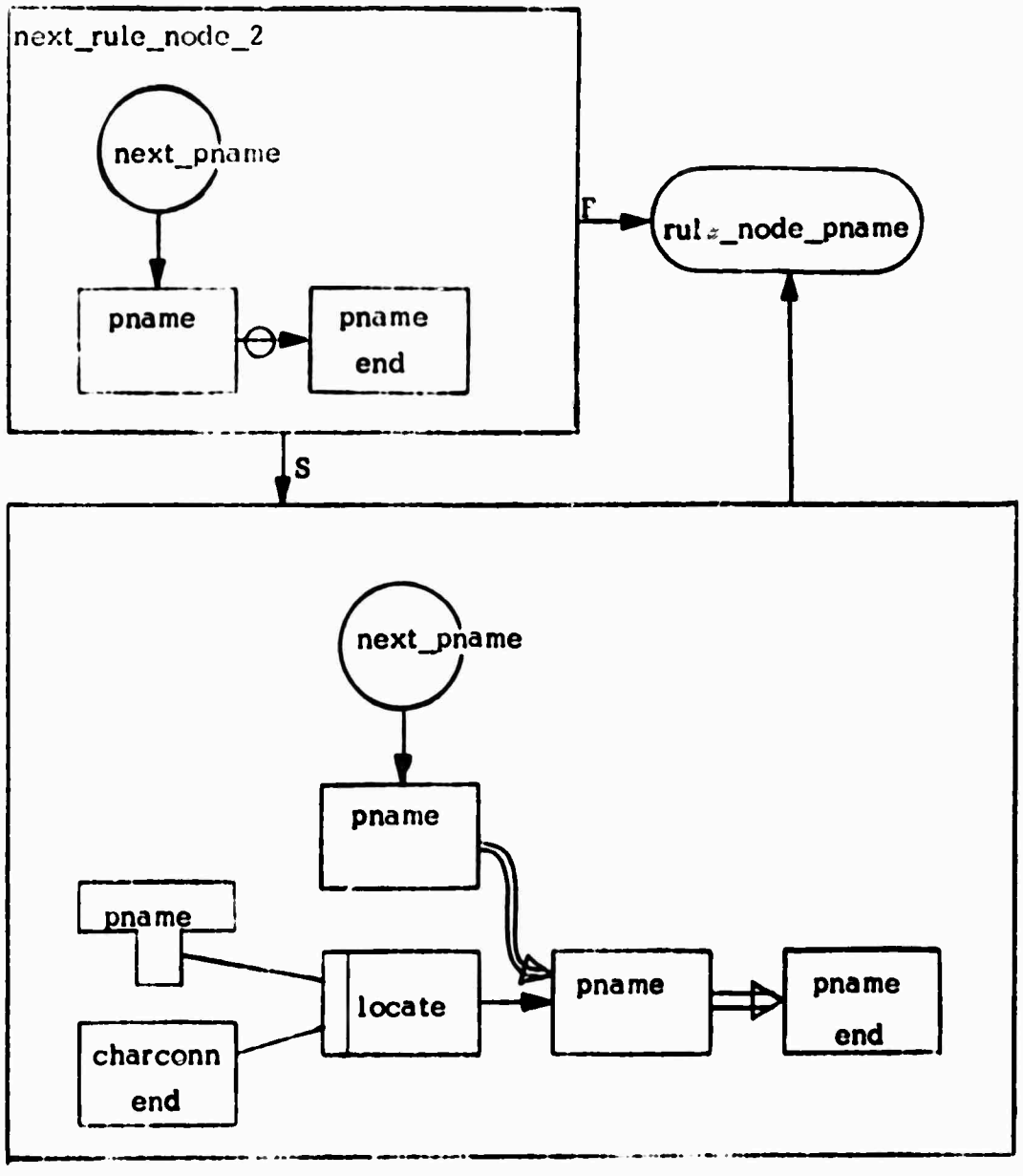
(cont' from previous page)

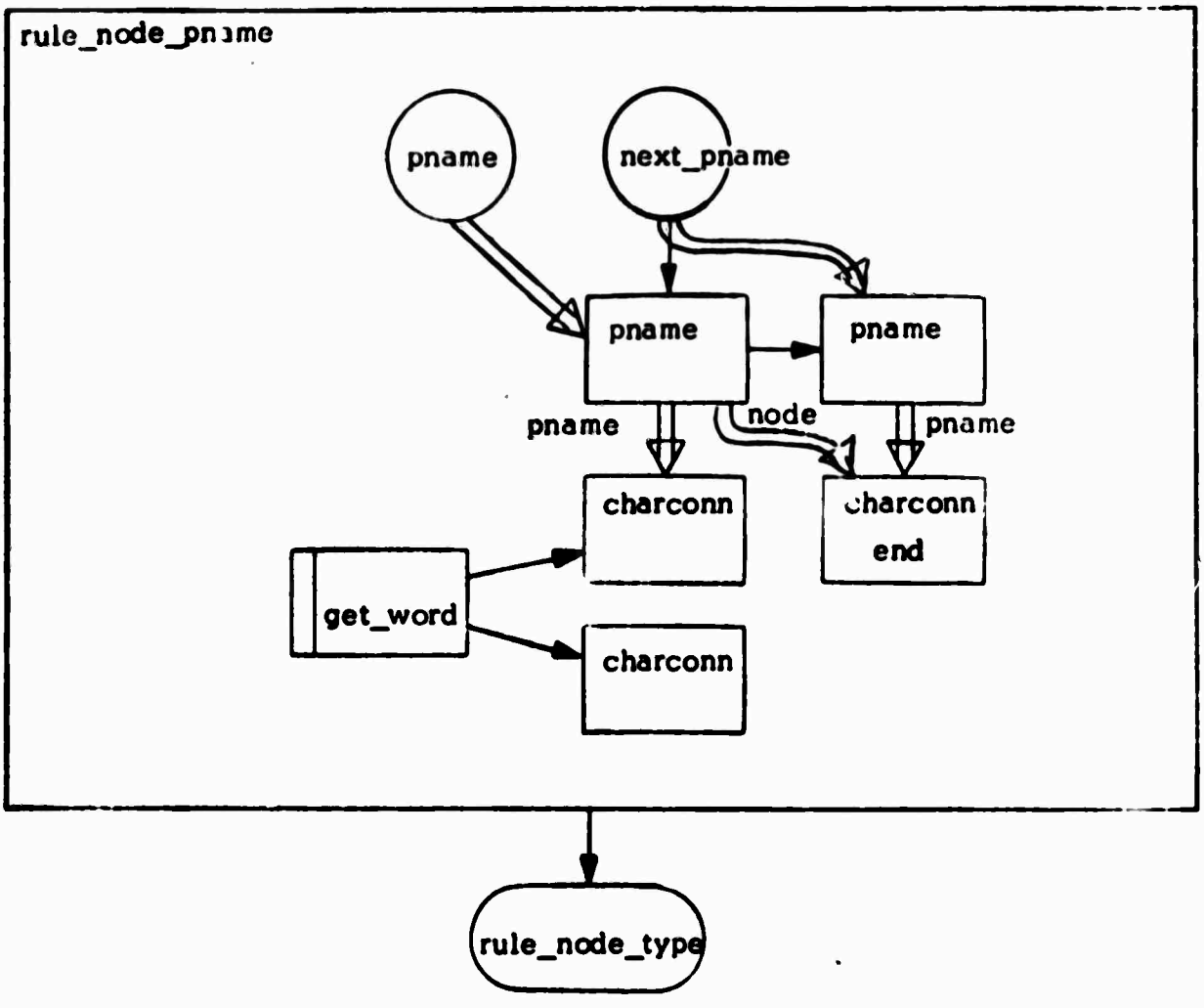


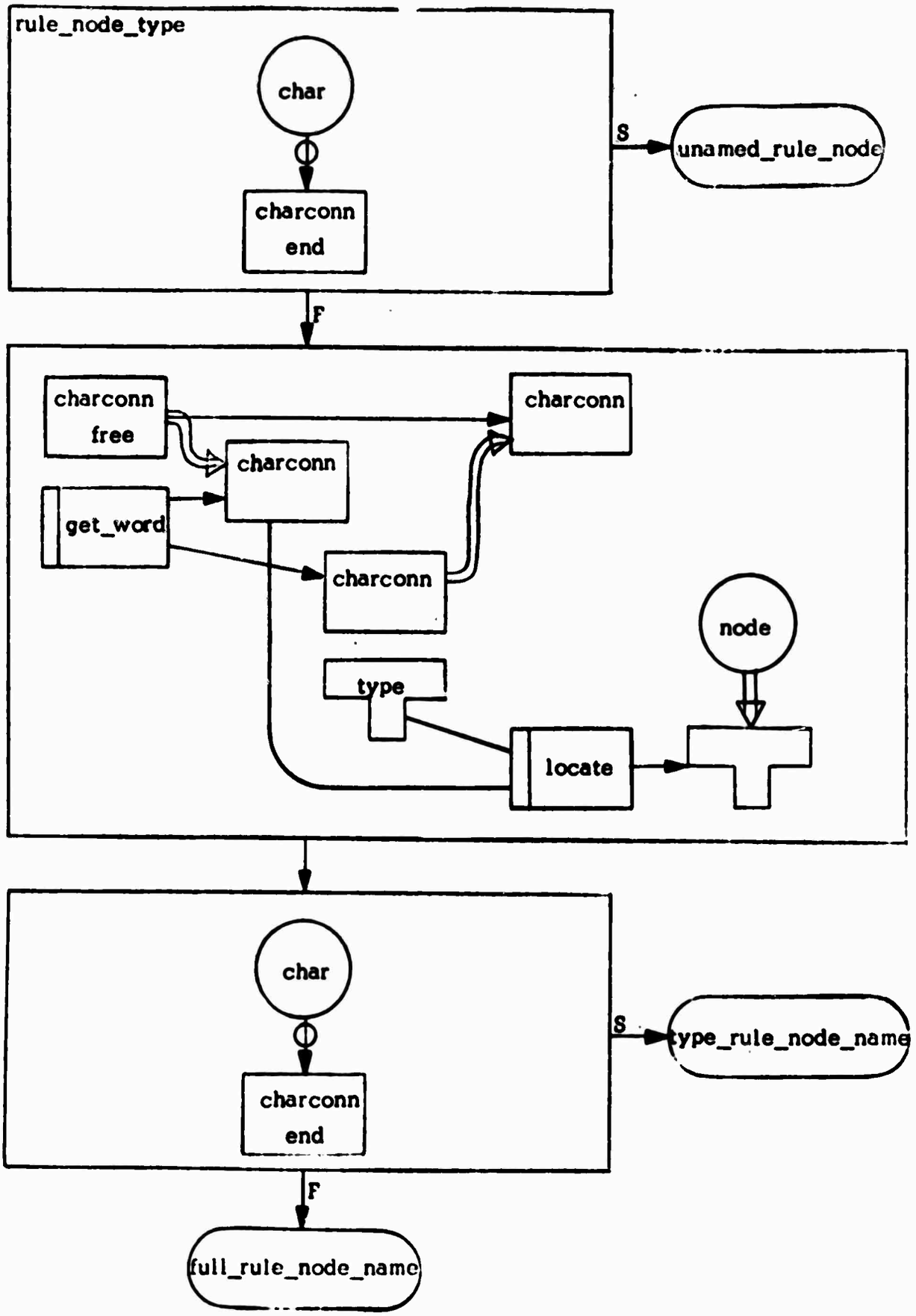

minus_rule
char
charconn
charconn
char
-
charconn
free
charconn
S
begin_rule
F
S
char
charconn
charconn
charconn
charconn
charconn
char
r
char
e
char
f
char
-
charconn
free
charconn
F
error
" ... " is an unknown statement.

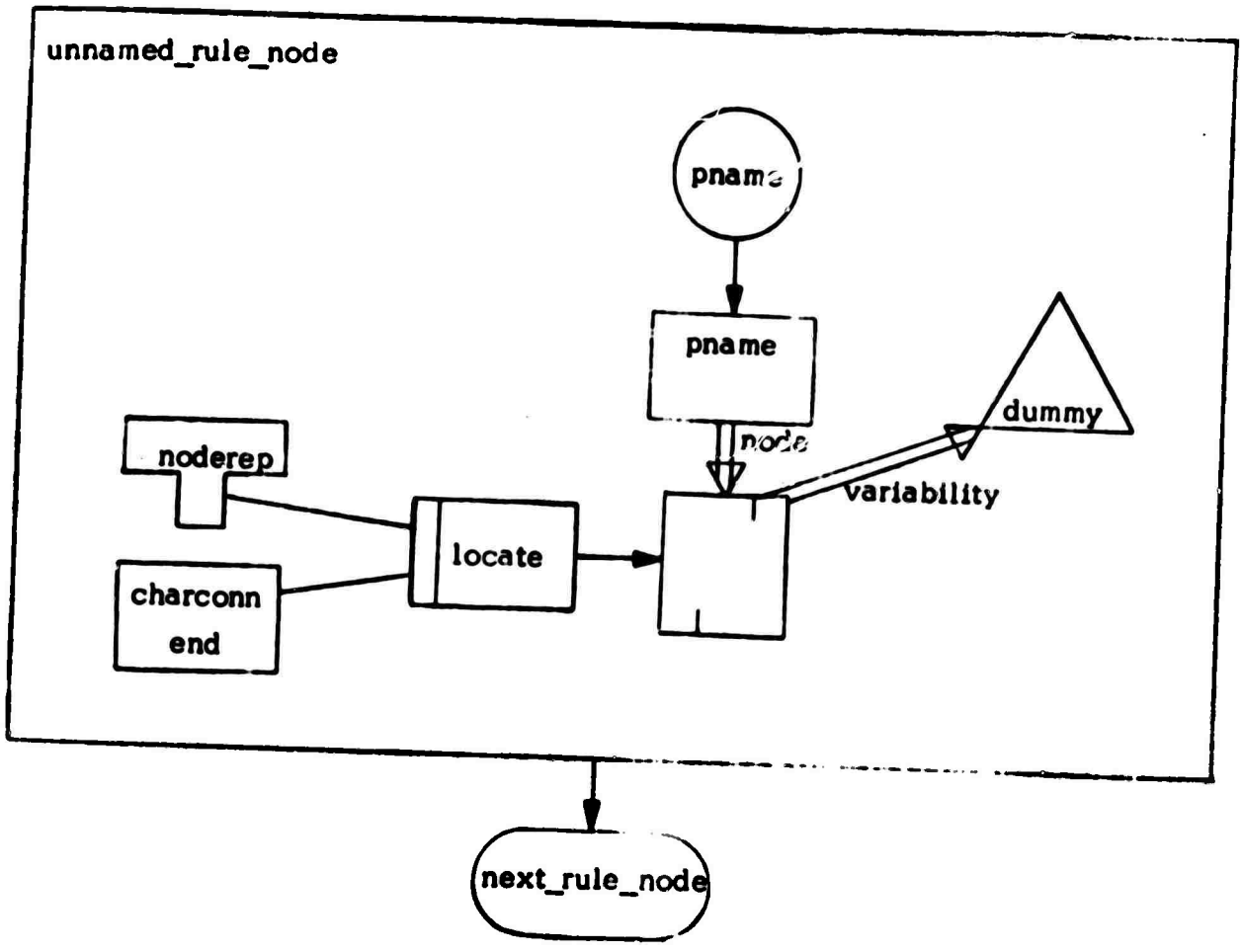

begin_rule
char
charconn
charconn
special
SP
charconn
free
charconn
SF
char
charconn
end
S
error
A page-name is missing for a "-rule-" or "-ruleref-" on a loader page.
F
pname
charconn
begin_rule_3
get_word
charconn

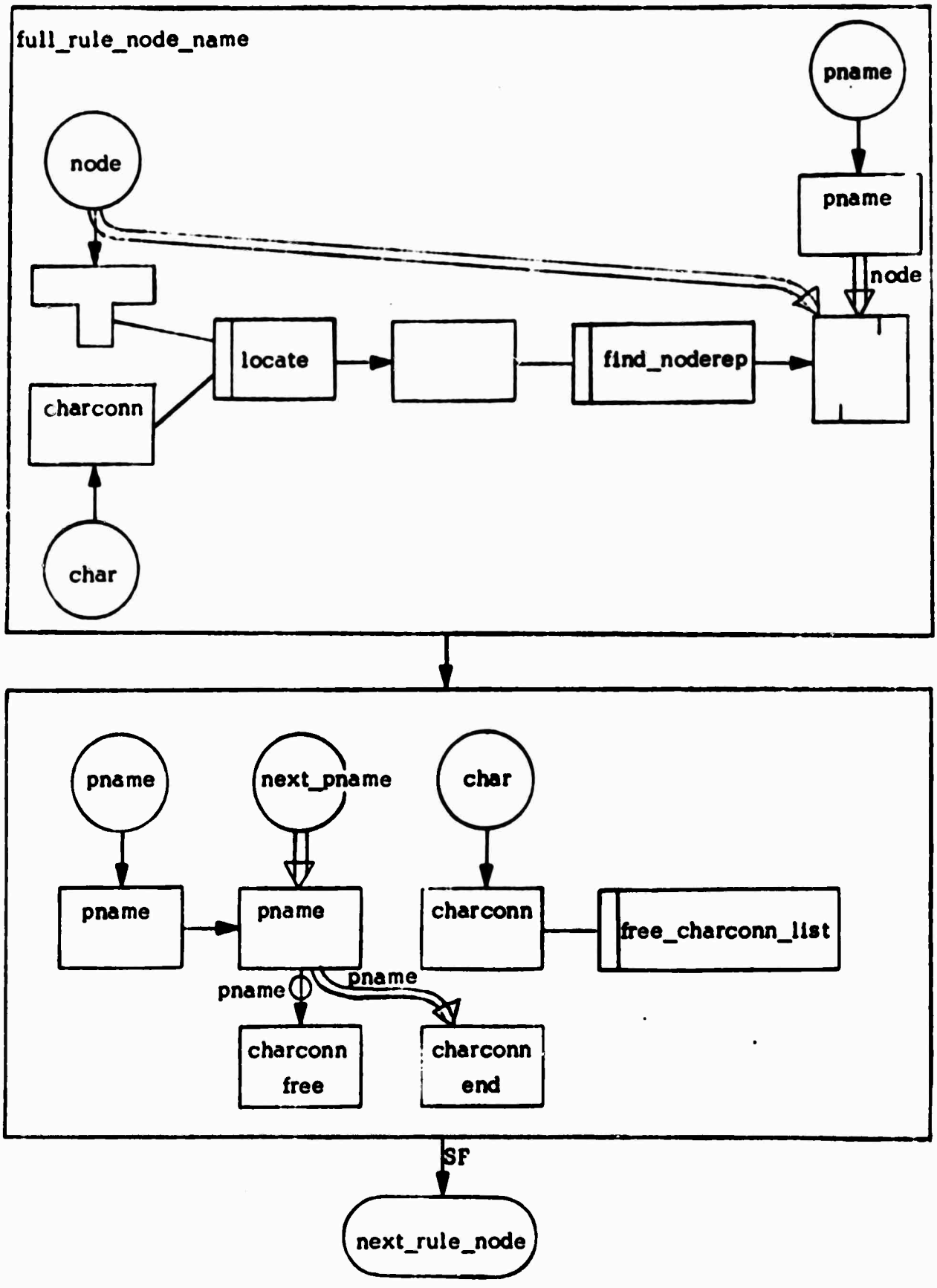

begin_rule_3
define_node
rule
rule
save_pname
pname_list
next_pname
char
-
pname
pname
S
F
next_node
node
clear
rule
state
contents
end
next_rule_node_2
minus_in_rule_nodes
F
S
next_rule_node
char
get_statement
charconn
char
charconn
char
-

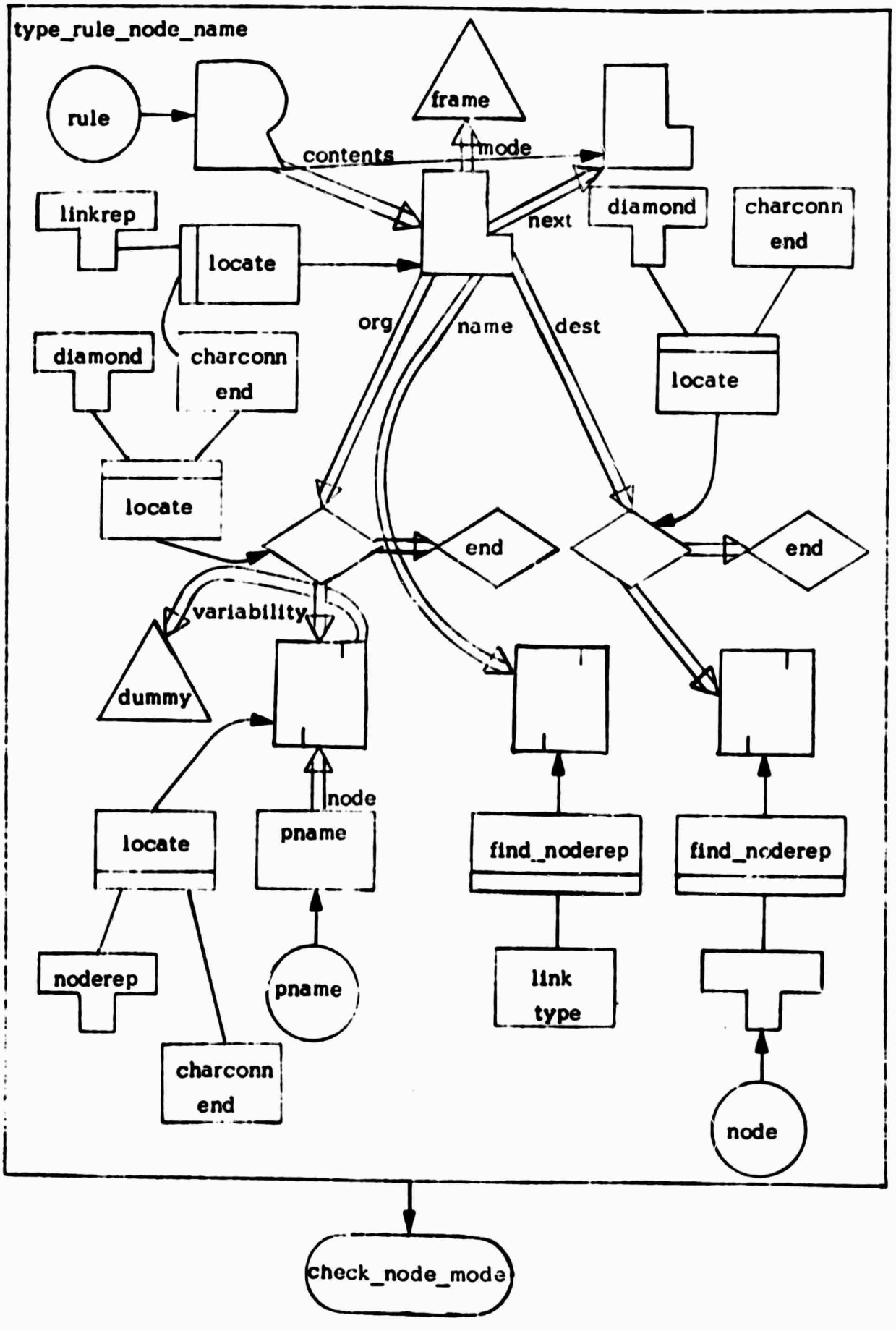

next_rule_node_2
next_pname
pname
pname
end
F
rule_node_pname
S
next_pname
pname
pname
locate
pname
pname
end
charconn
end

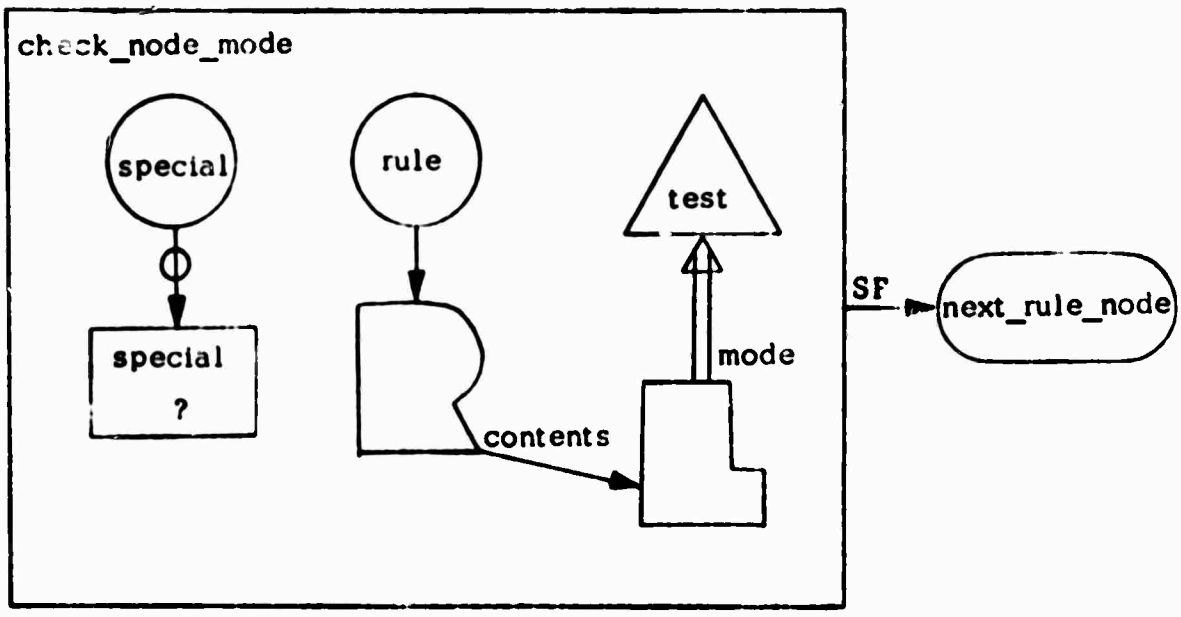

rule_node_pname
pname
next_pname
pname
pname
pname
node
pname
charconn
charconn
end
get_word
charconn
rule_node_type

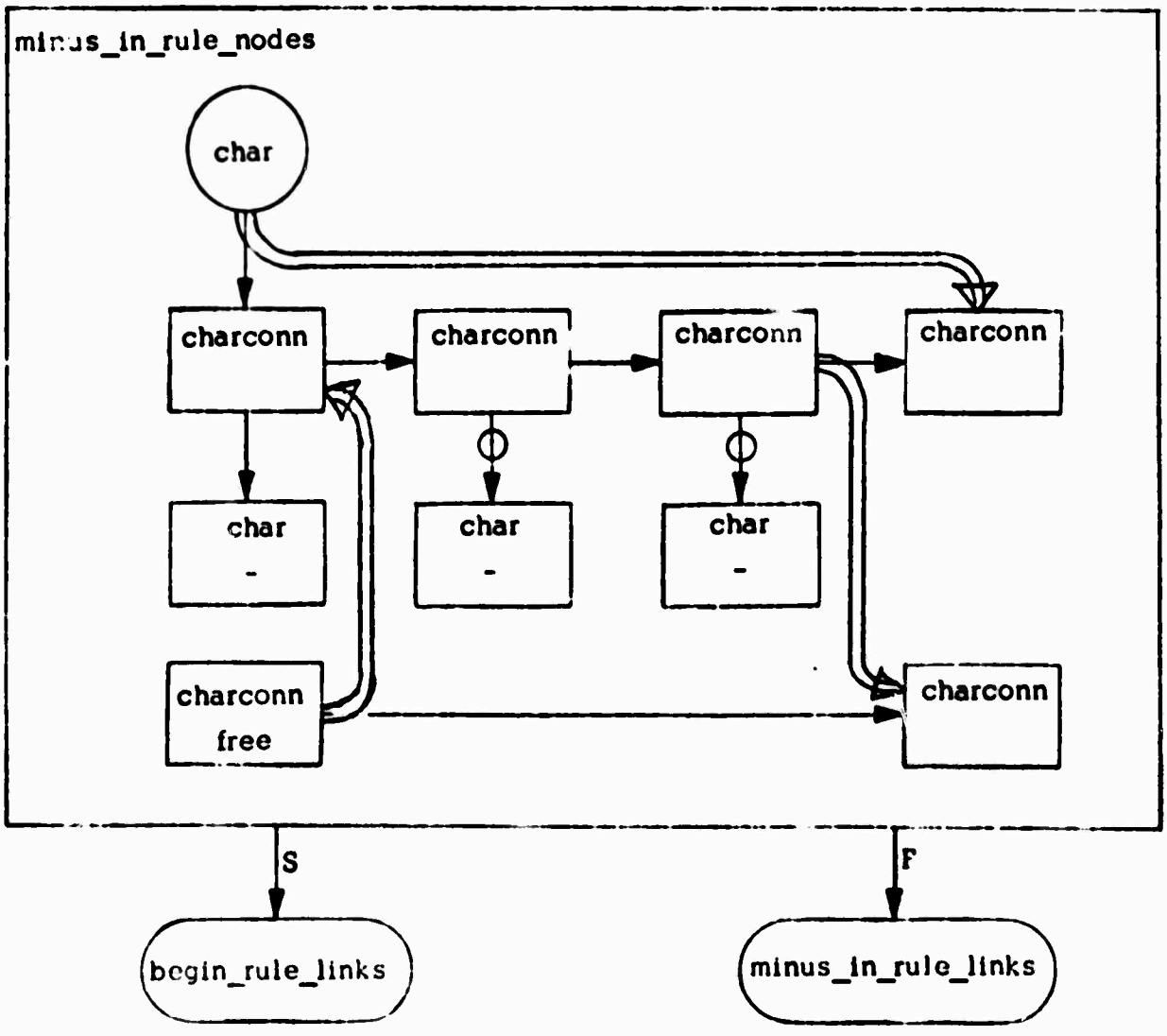

rule_node_type
char
charconn
end
S
unamed_rule_node
F
charconn
free
charconn
charconn
get_word
charconn
node
type
locate
char
charconn
end
S
type_rule_node_name
F
full_rule_node_name

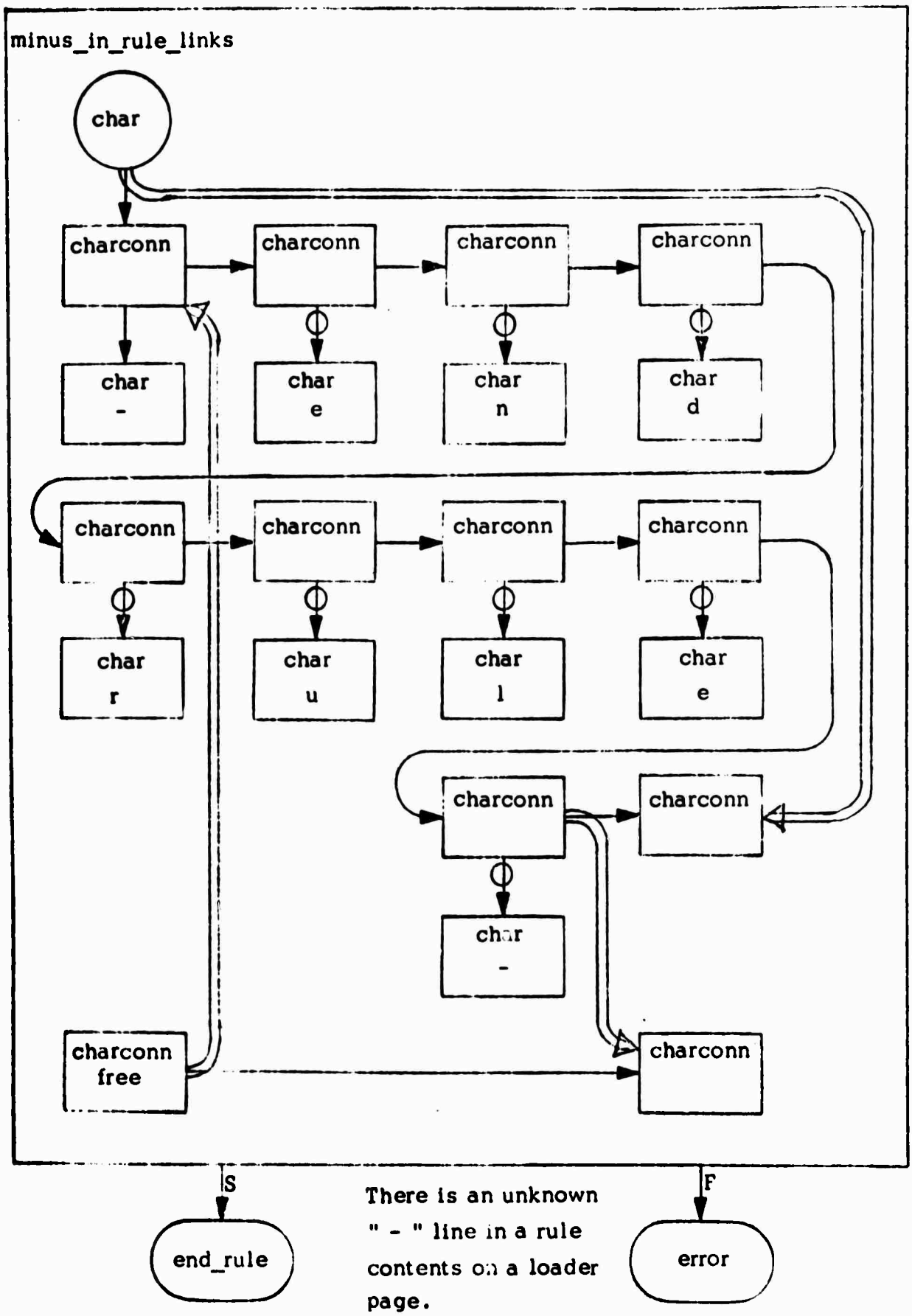

unnamed_rule_node
pname
pname
node
dummy
variability
noderep
charconn
end
locate
next_rule_node


full_rule_node_name
pname
node
pname
node
locate
find_noderep
charconn
char
pname
next_pname
char
pname
pname
charconn
free_charconn_list
pname
pname
charconn
free
charconn
end
SF
next_rule_node

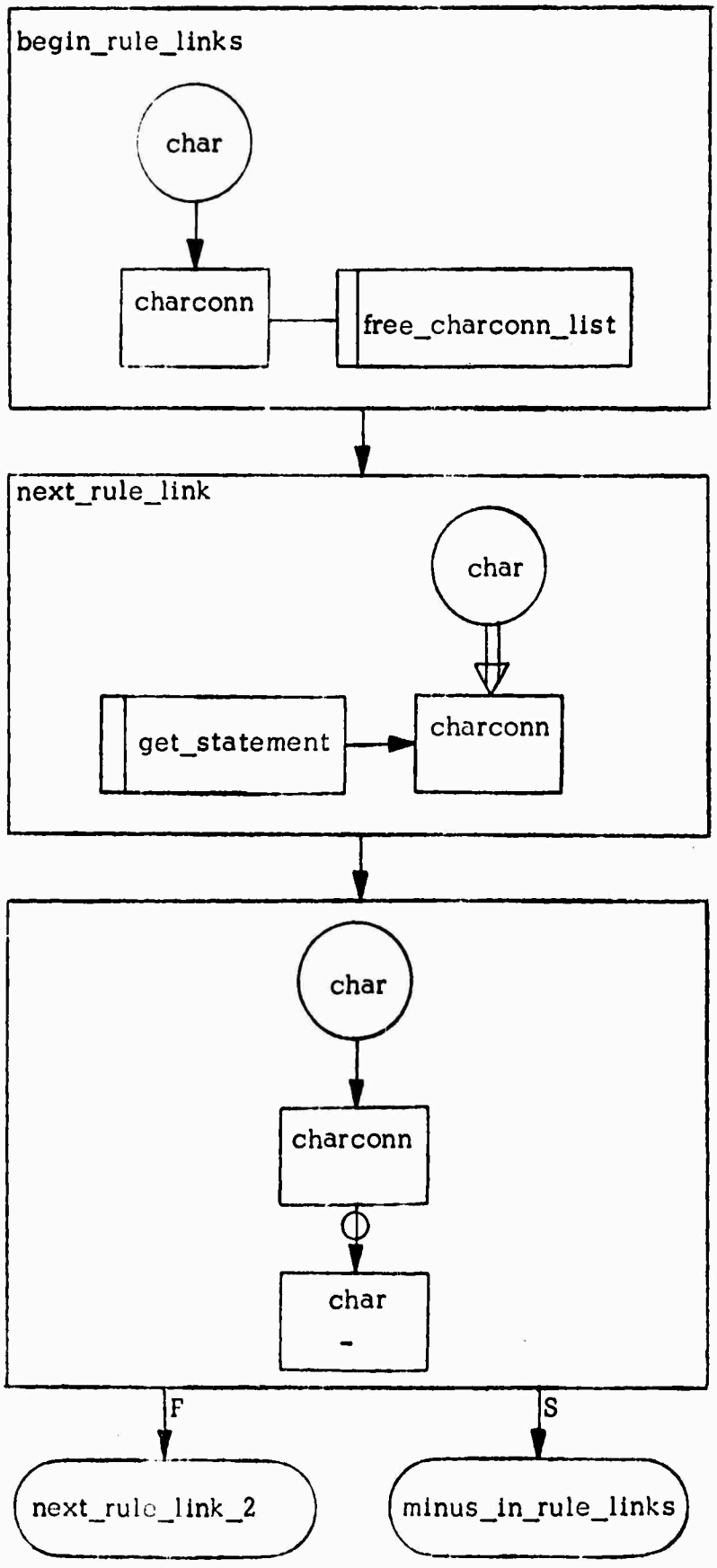

type_rule_node_name
rule
frame
contents
mode
linkrep
locate
next
diamond
charconn
end
org
name
dest
diamond
charconn
end
locate
locate
end
end
variability
dummy
node
locate
pname
find_noderep
find_noderep
noderep
pname
link
type
charconn
end
node
check_node_mode

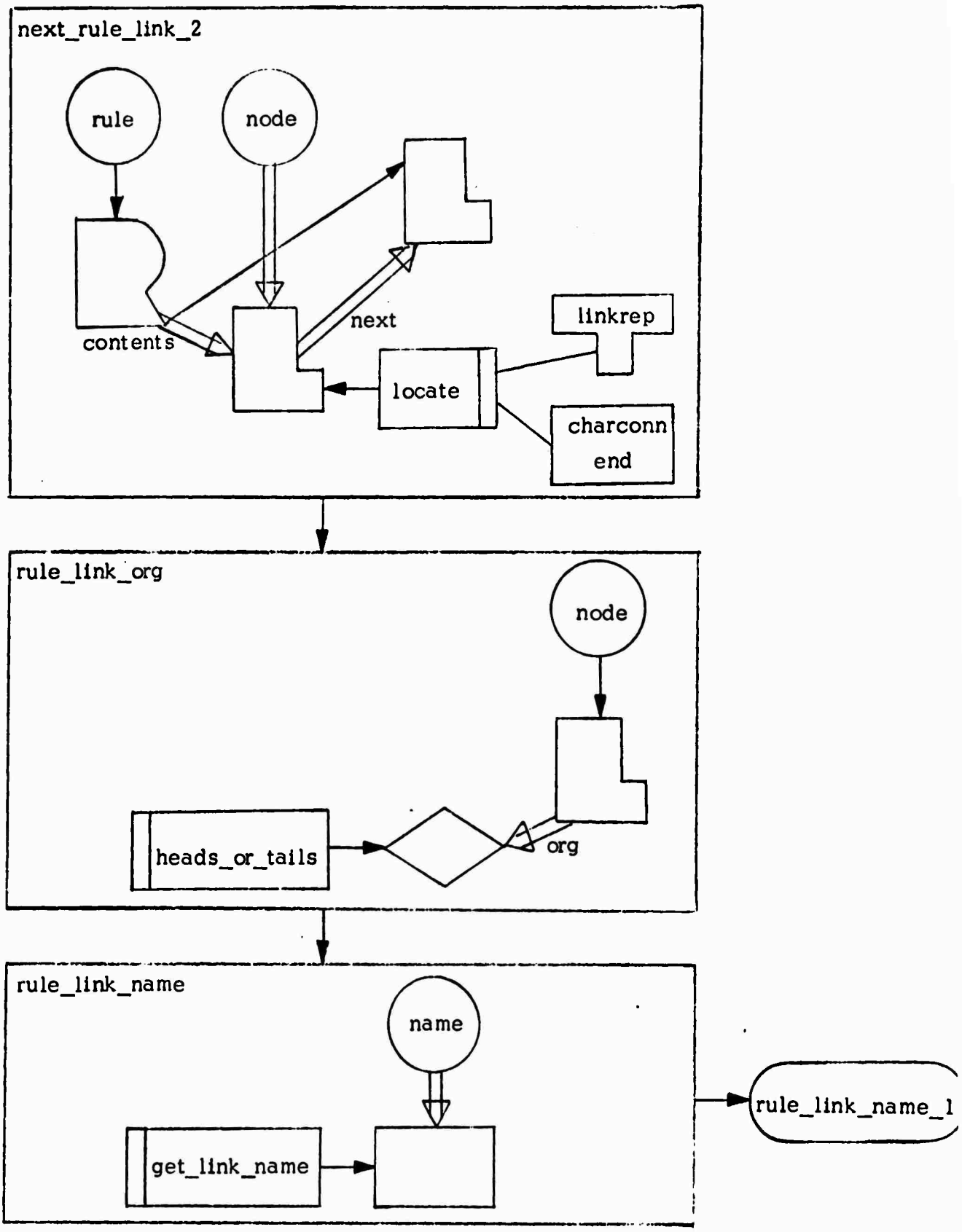

check_node_mode
special
special
?
rule
test
mode
contents
SF
next_rule_node

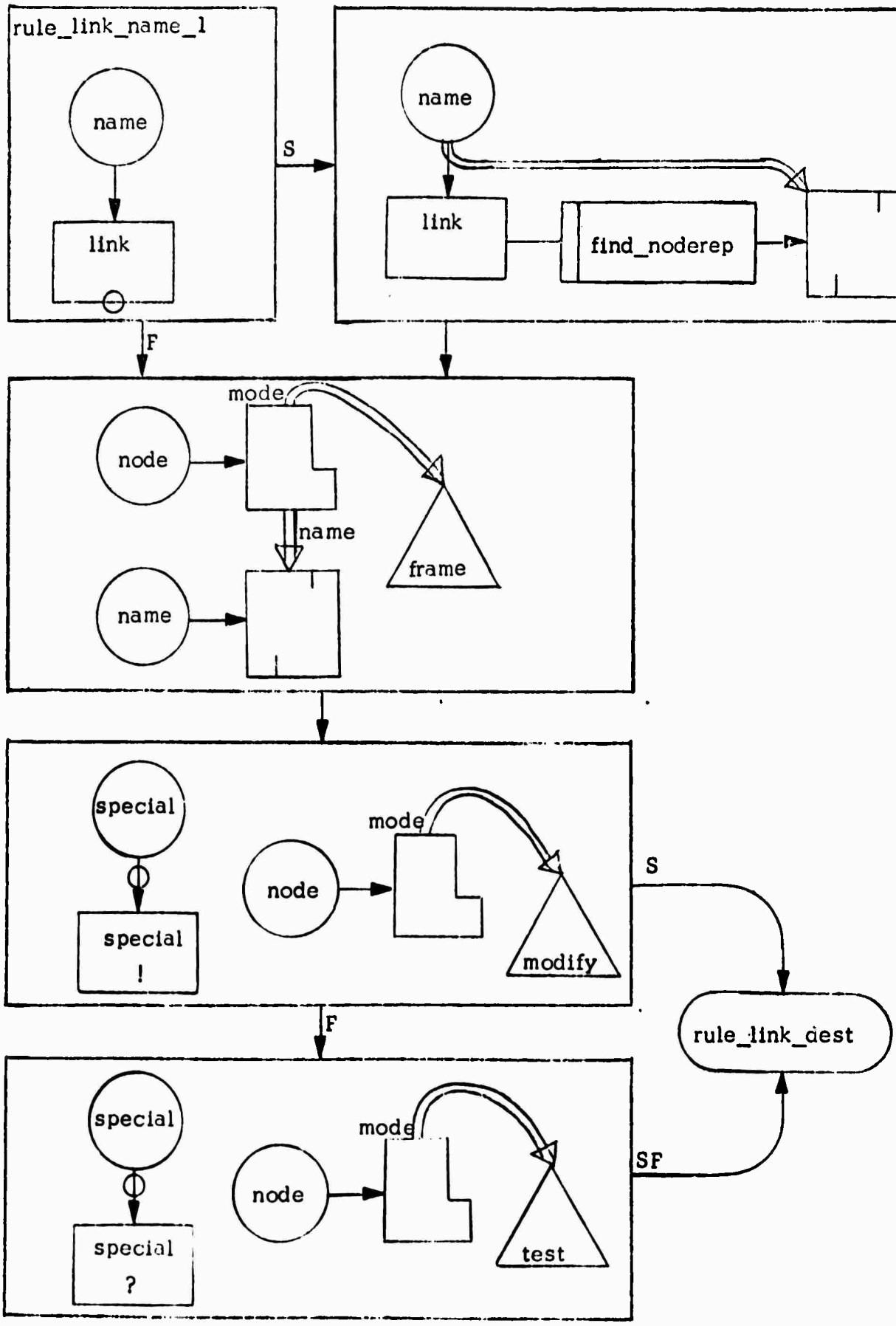

minus_in_rule_nodes
char
charconn
charconn
charconn
charconn
char
-
char
-
char
-
charconn
free
charconn
S
F
begin_rule_links
minus_in_rule_links

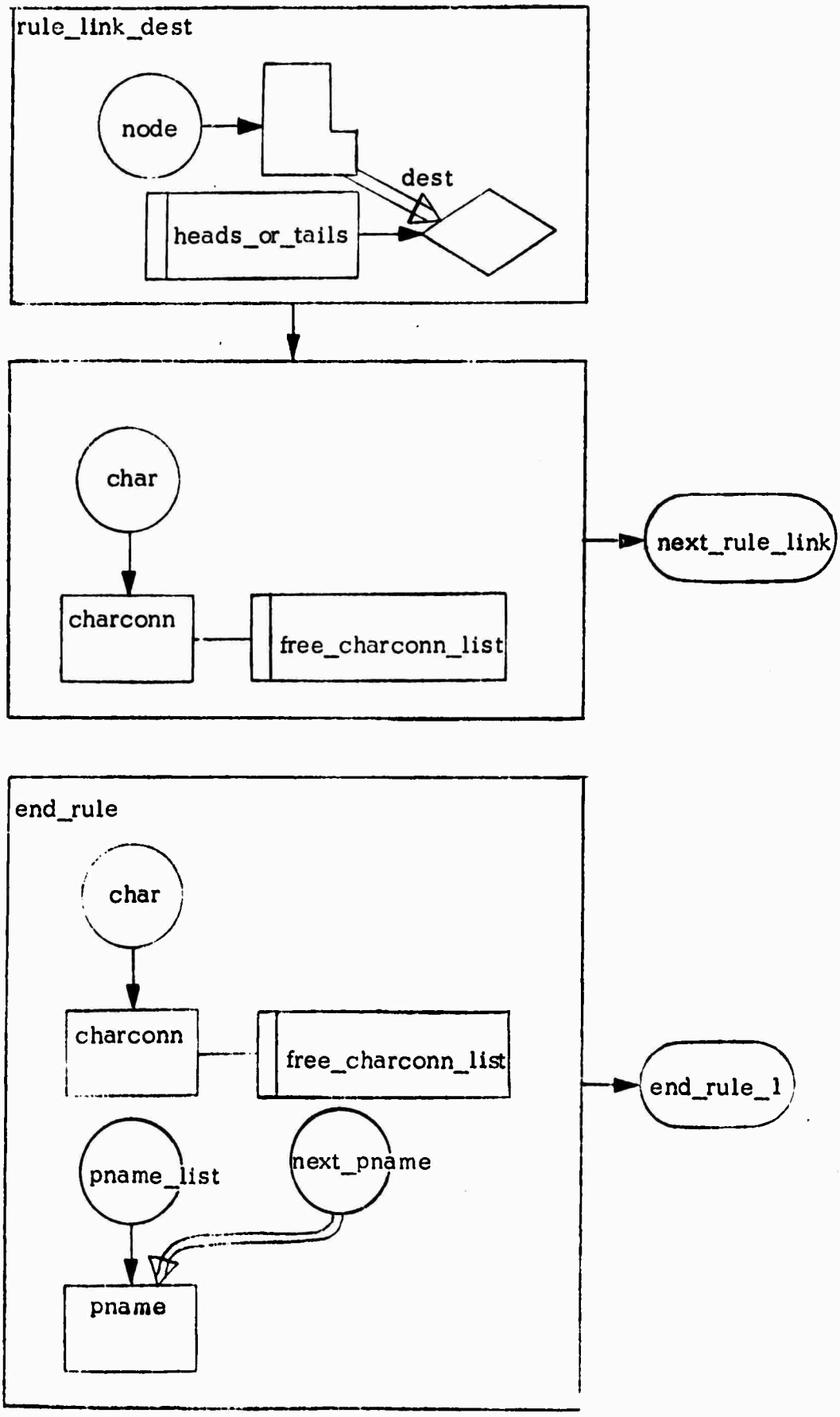


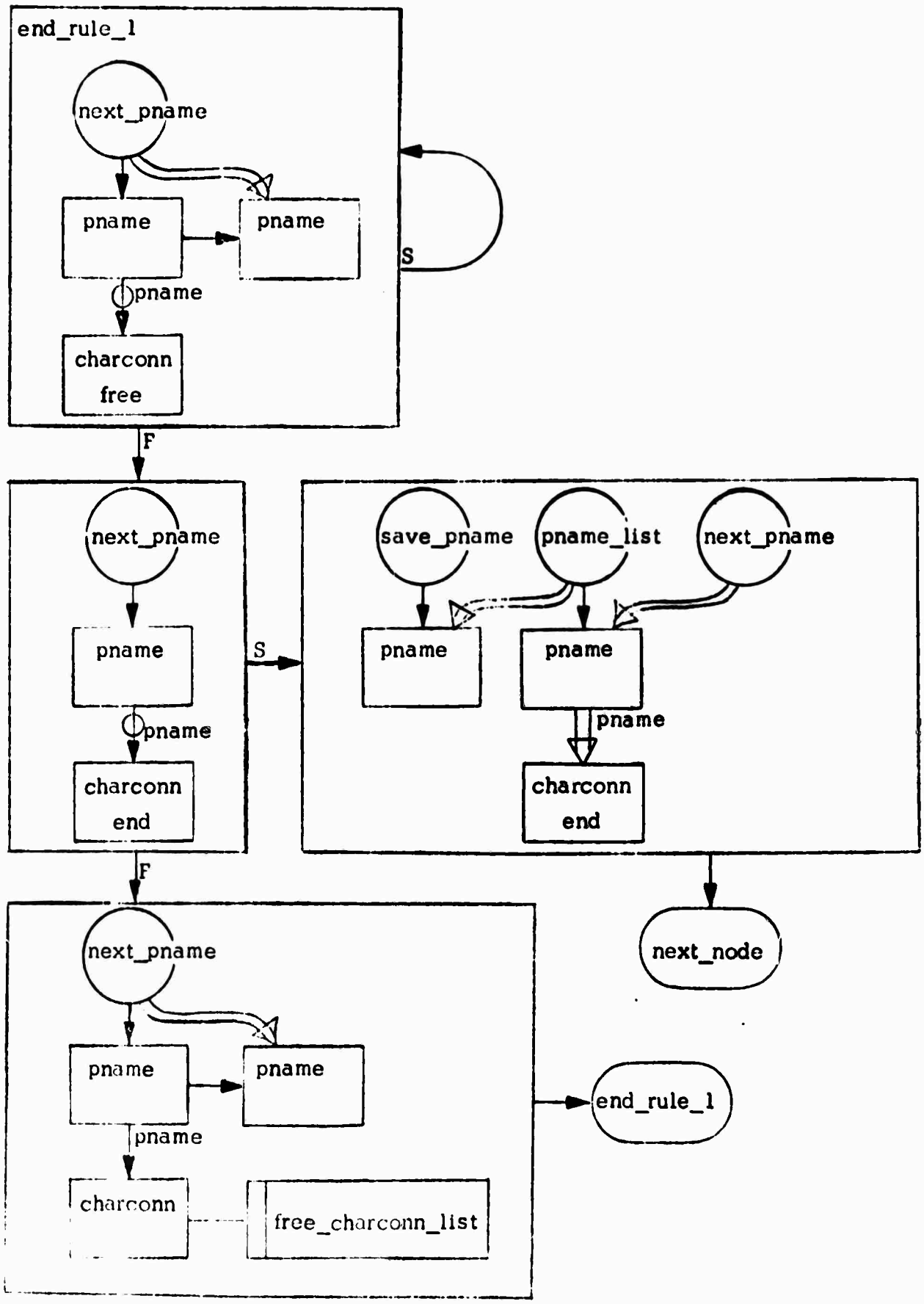
There is an unknown " - " line in a rule contents on a loader page.

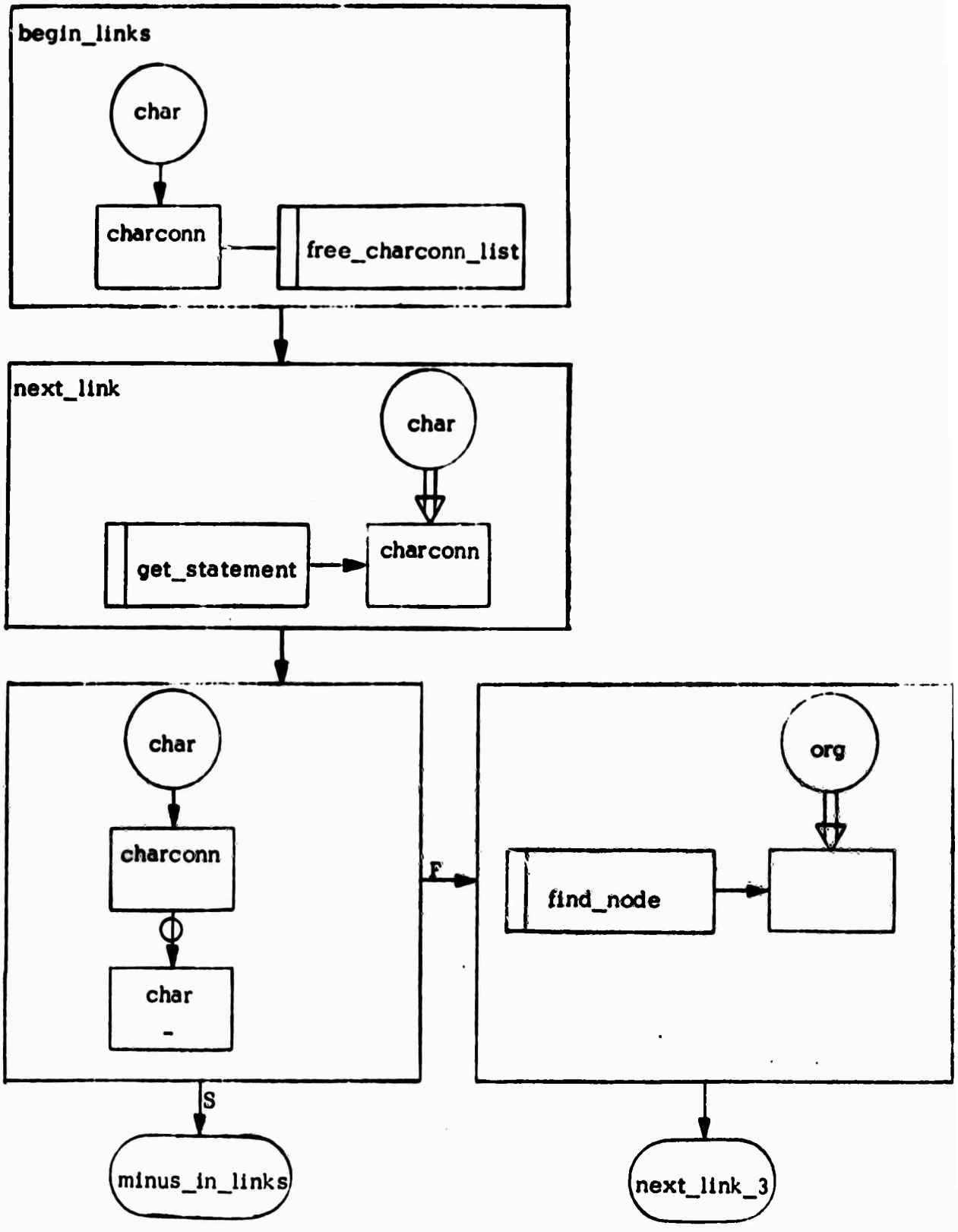

begin_rule_links
char
charconn
free_charconn_list
next_rule_link
char
get_statement
charconn
char
charconn
char
-
F
S
next_rule_link_2
minus_in_rule_links

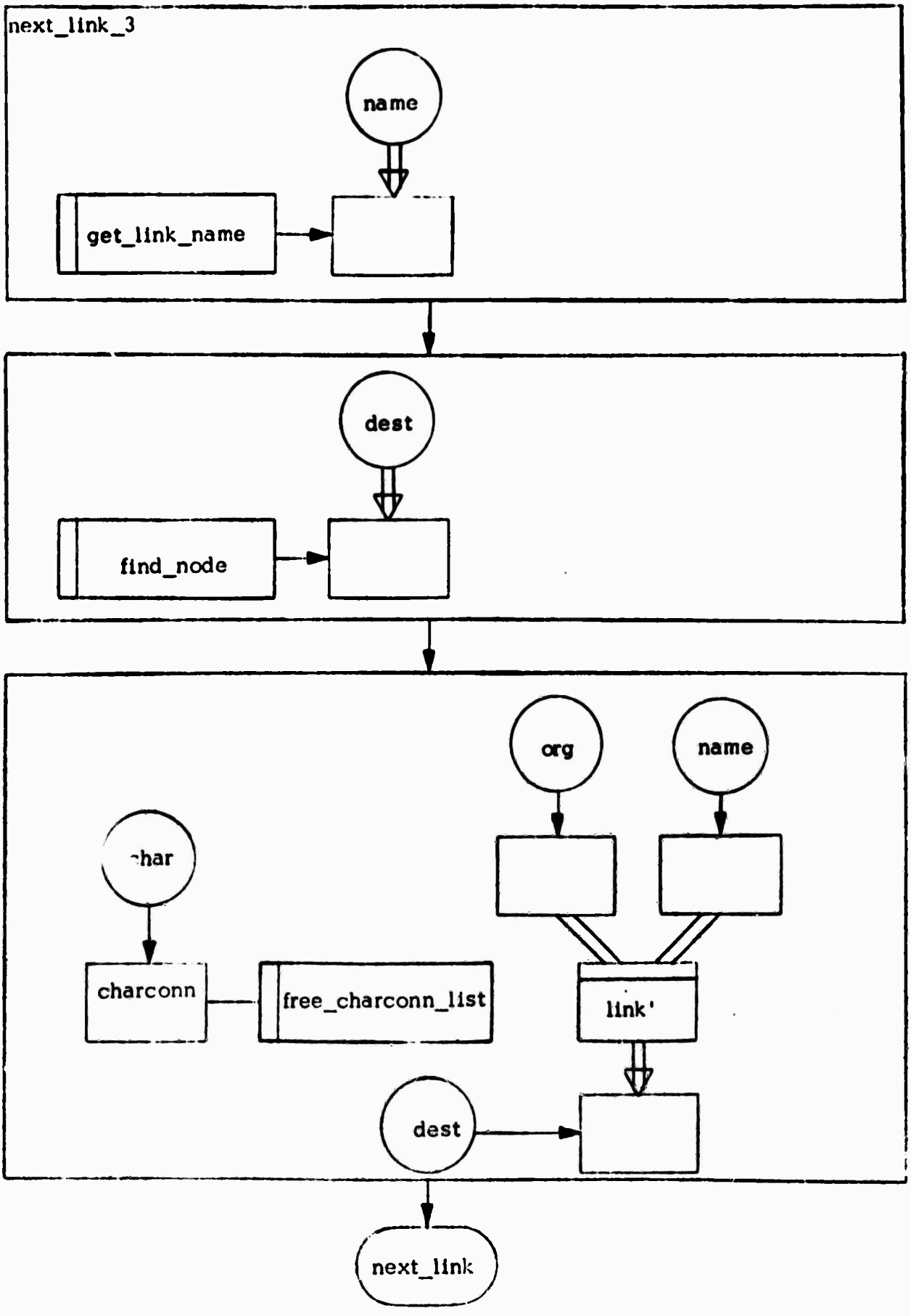

next_rule_link_2
rule
node
contents
next
linkrep
locate
charconn
end
rule_link_org
node
heads_or_tails
org
rule_link_name
name
get_link_name
rule_link_name_1

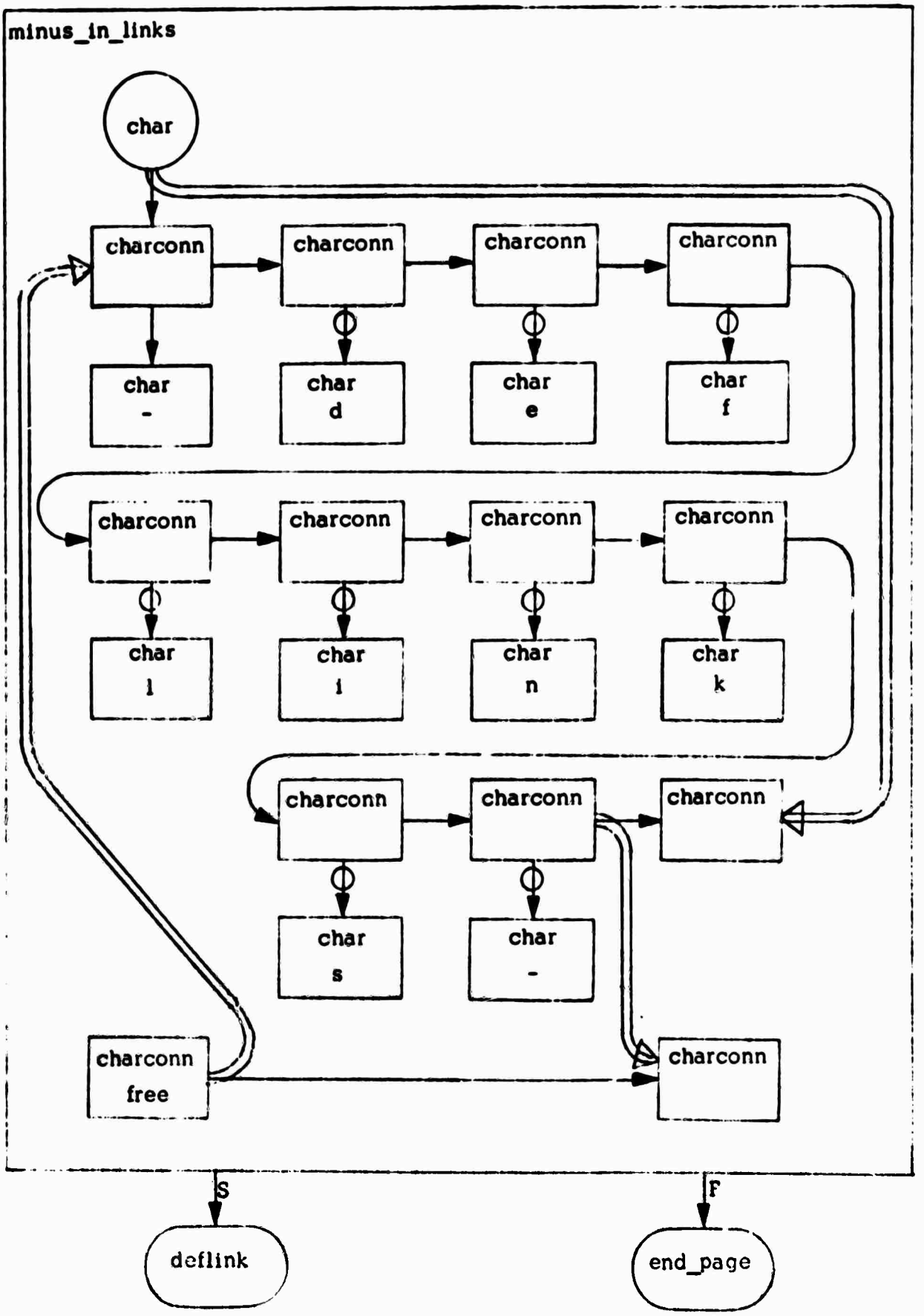

rule_link_name_1
name
link
S
name
link
find_noderep
F
mode
node
name
frame
name
special
special
!
mode
node
modify
S
F
rule_link_dest
special
special
?
mode
node
test
SF

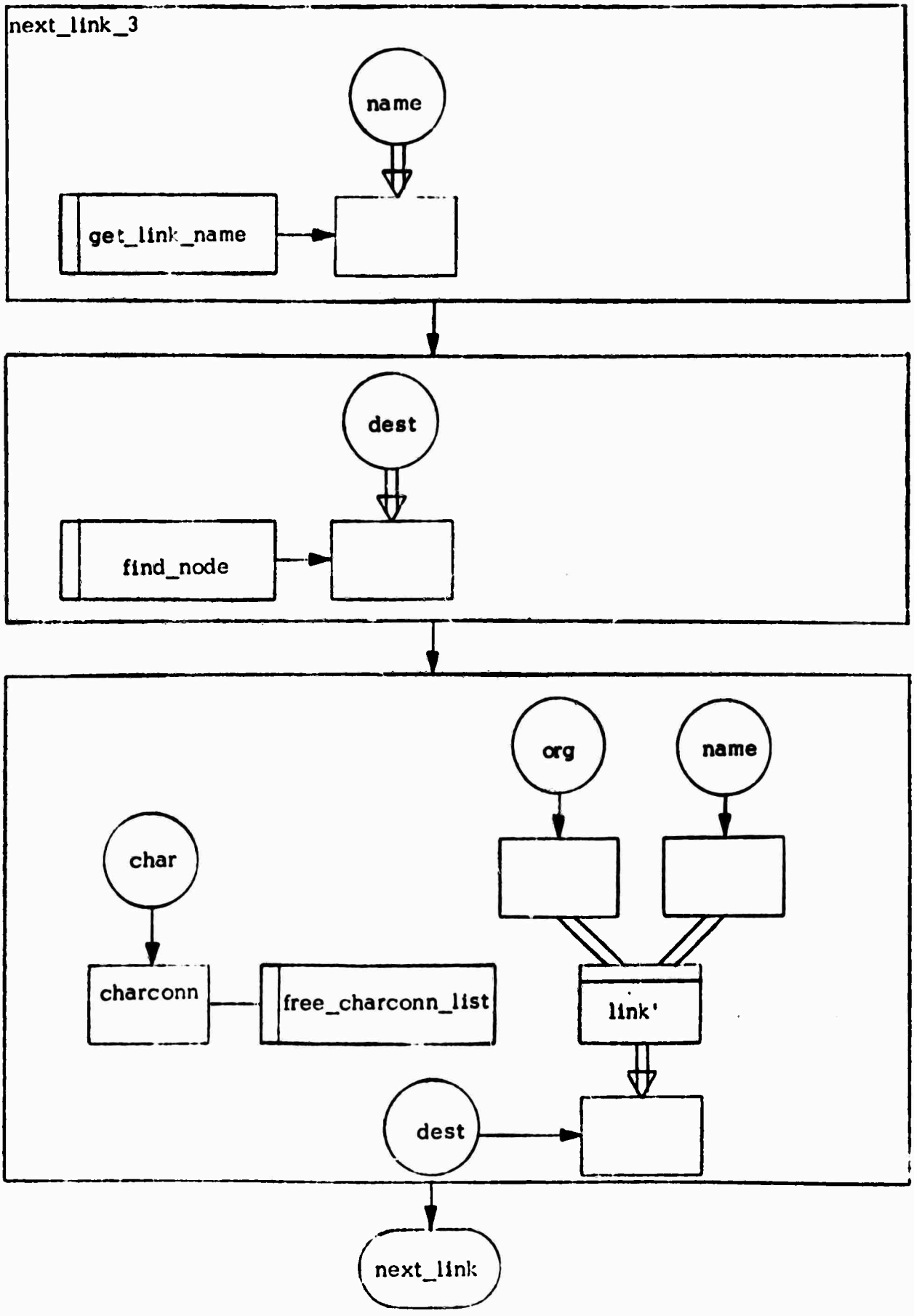

rule_link_dest
node
dest
heads_or_tails
char
charconn
free_charconn_list
next_rule_link
end_rule
char
charconn
free_charconn_list
end_rule_1
pname_list
next_pname
pname

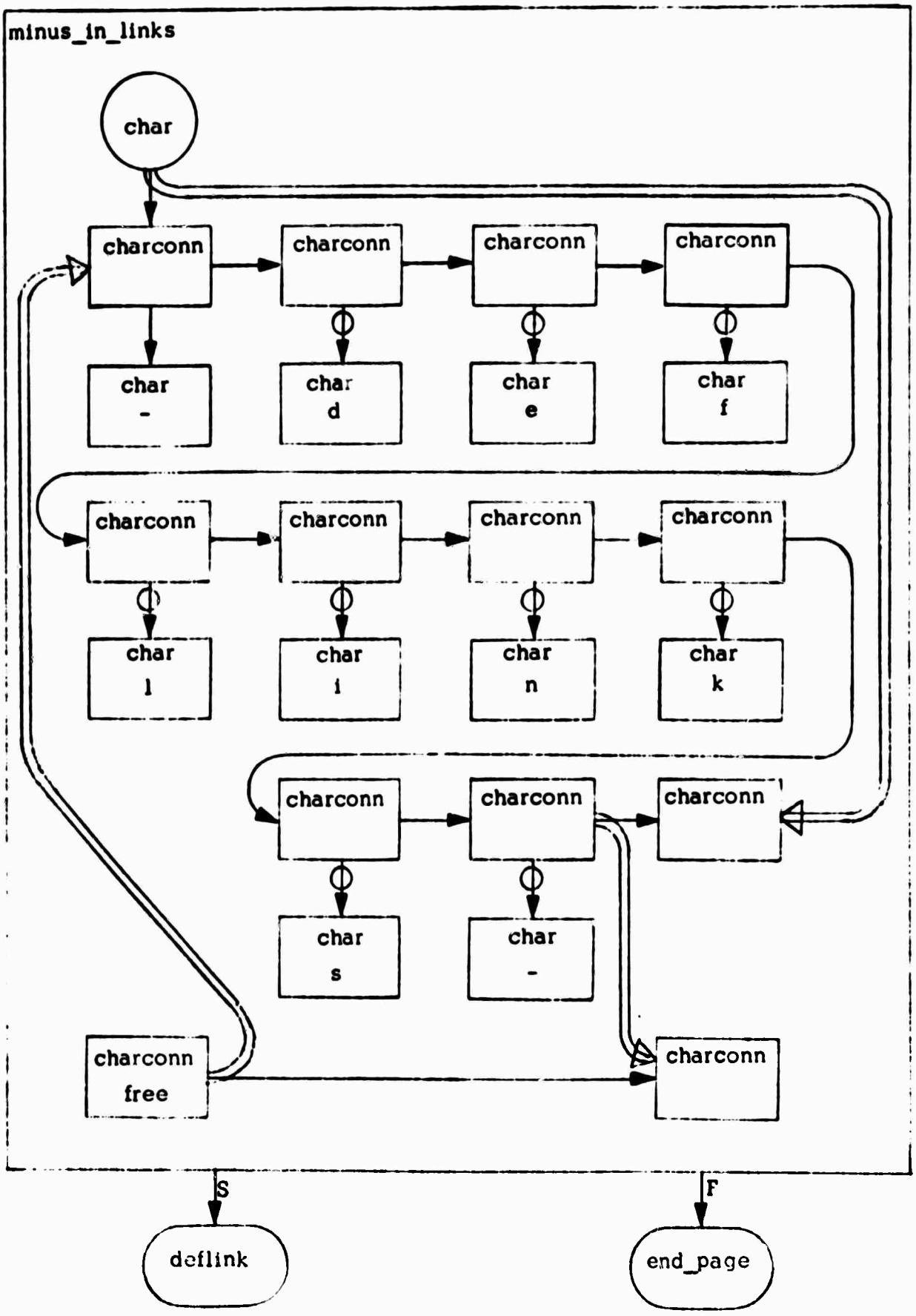

end_rule_1
next_pname
pname
pname
pname
charconn
free
S
F
next_pname
pname
pname
charconn
end
S
save_pname
pname_list
next_pname
pname
pname
pname
charconn
end
F
next_node
next_pname
pname
pname
pname
charconn
free_charconn_list
end_rule_1

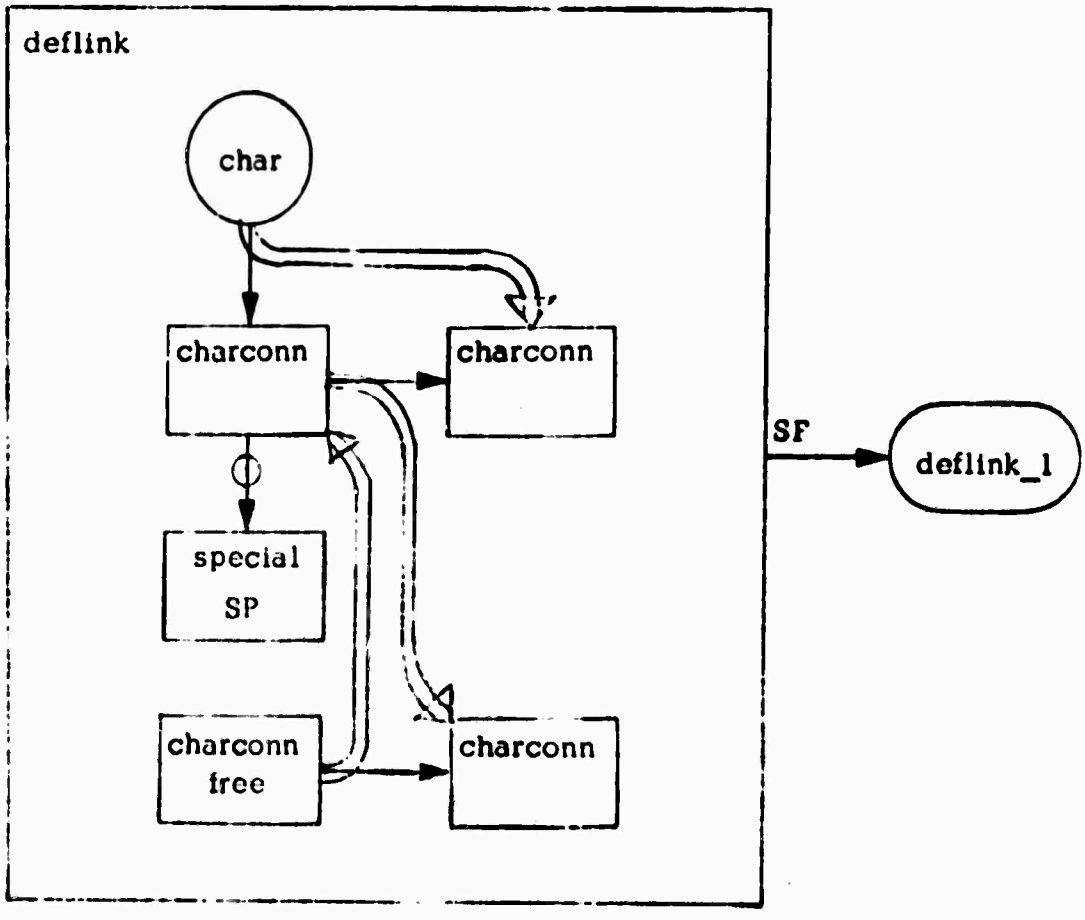

begin_links
char
charconn
free_charconn_list
next_link
char
get_statement
charconn
char
charconn
char
-
S
F
org
find_node
minus_in_links
next_link_3

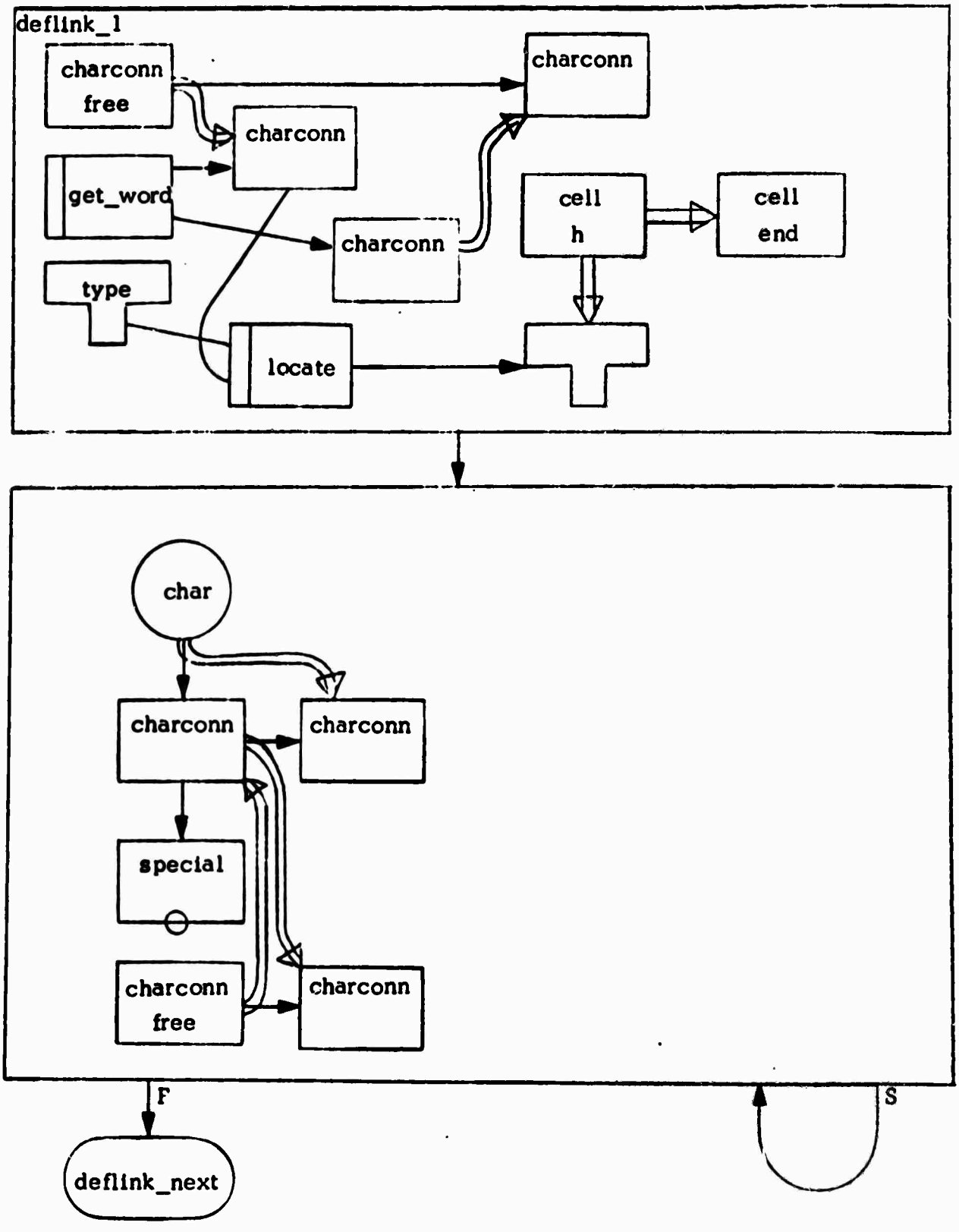

next_link_3
name
get_link_name
dest
find_node
org
name
char
charconn
free_charconn_list
link'
dest
next_link

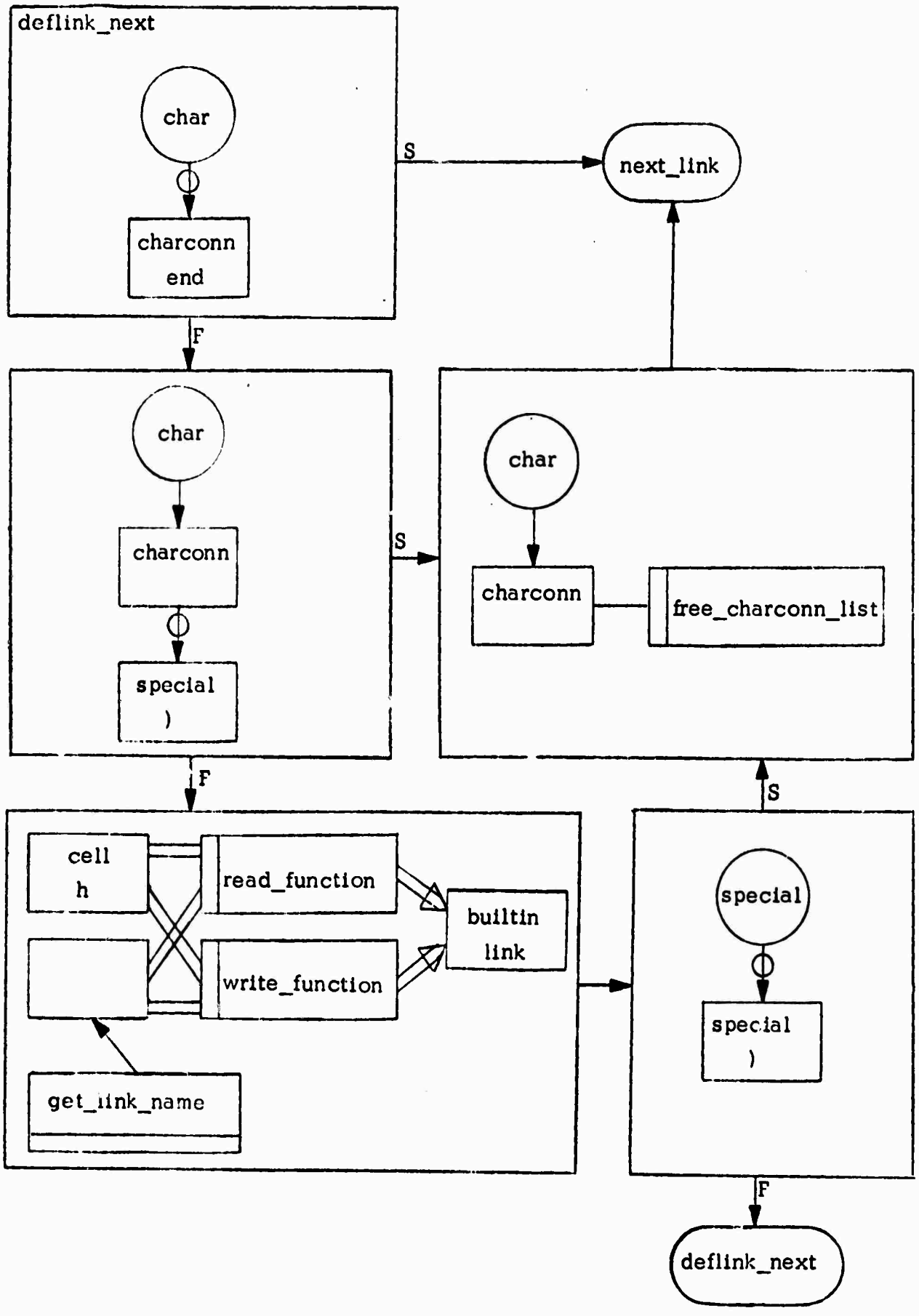

minus_in_links
char
charconn
charconn
charconn
charconn
char
-
char
d
char
e
char
f
charconn
charconn
charconn
charconn
char
l
char
i
char
n
char
k
charconn
charconn
charconn
char
s
char
-
charconn
free
charconn
S
F
deflink
end_page

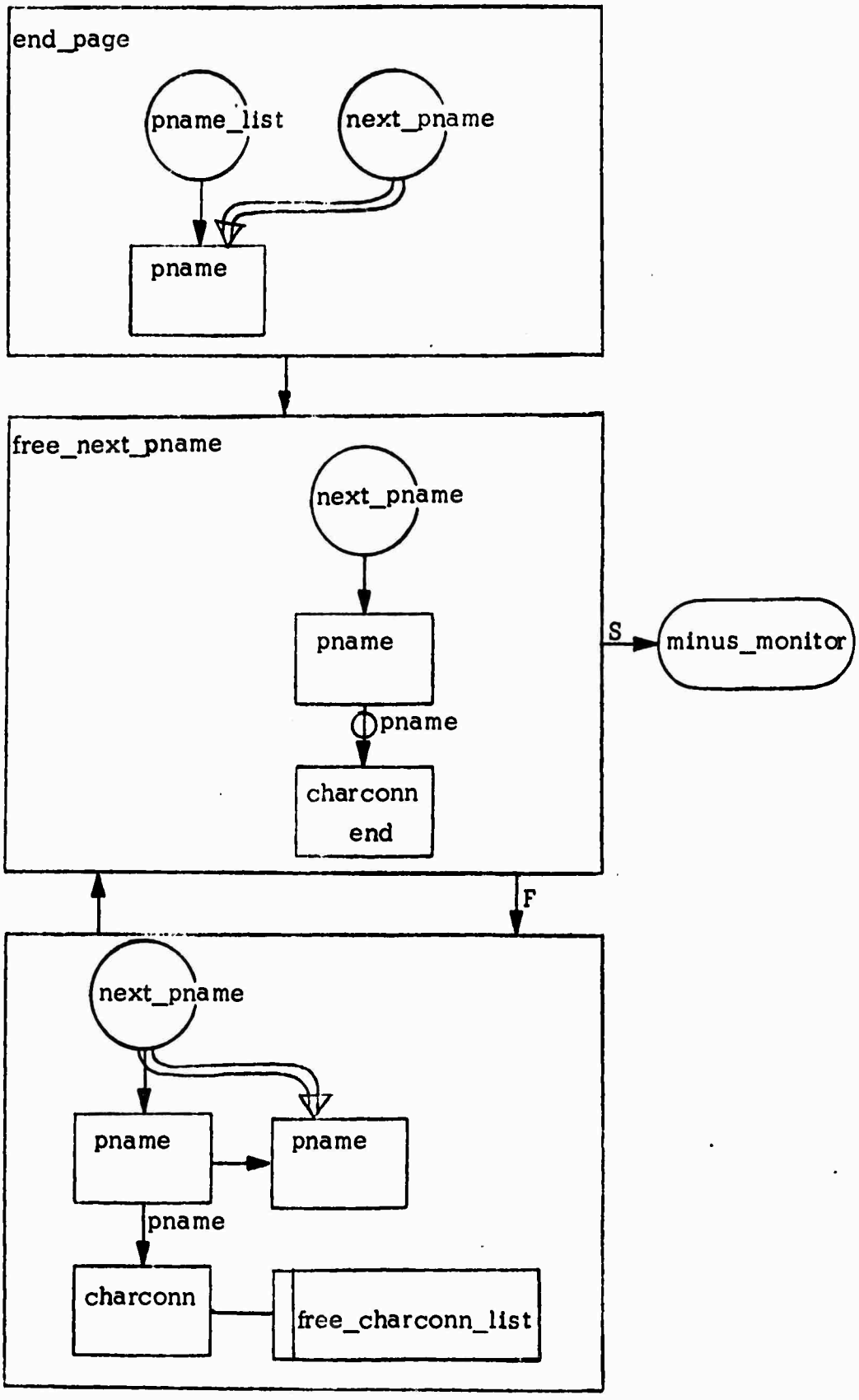

next_link_3
name
get_link_name
dest
find_node
org
name
char
charconn
free_charconn_list
link'
dest
next_link

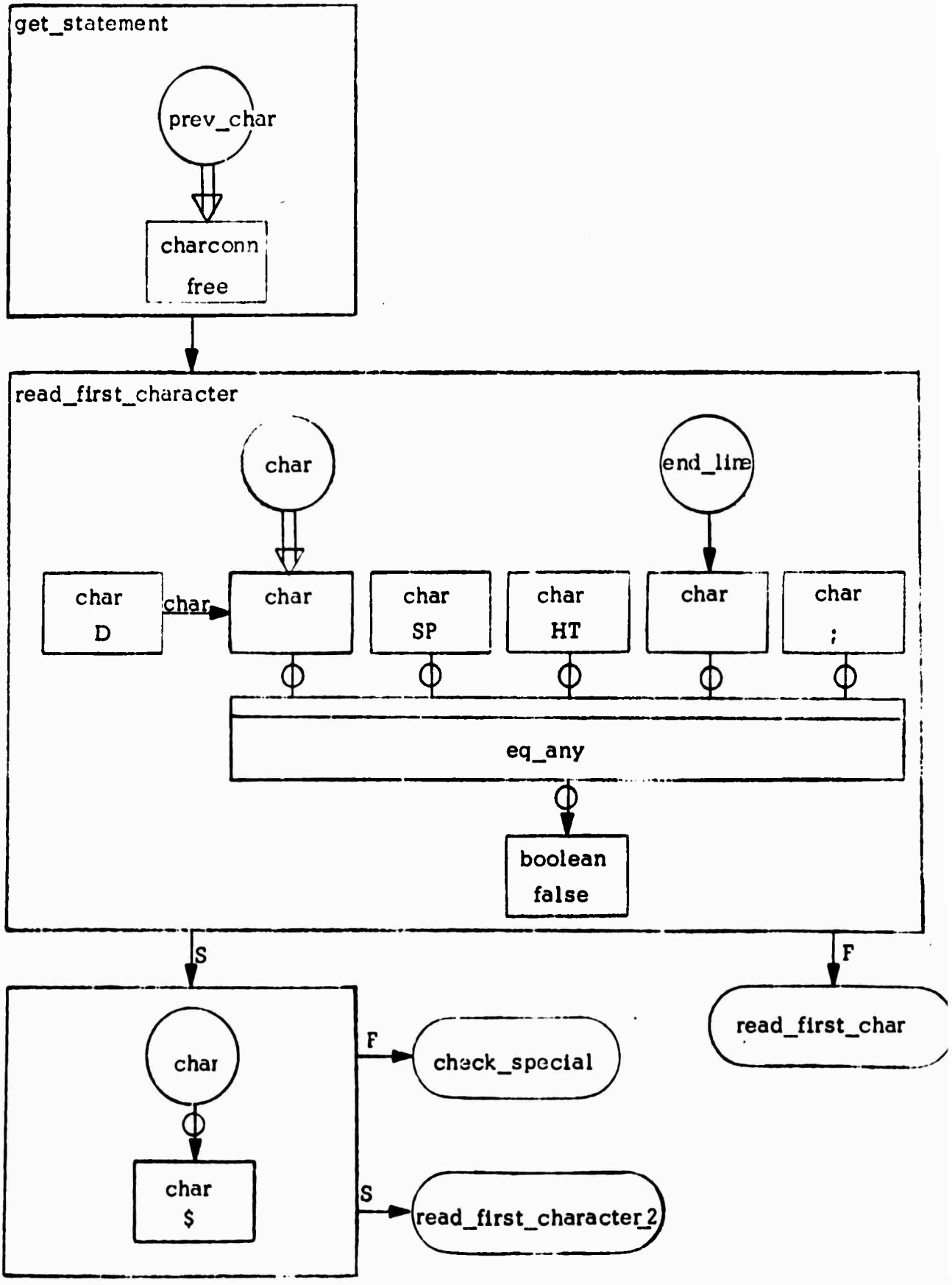

minus_in_links
char
charconn
charconn
charconn
charconn
char
-
char
d
char
e
char
f
charconn
charconn
charconn
charconn
char
l
char
i
char
n
char
k
charconn
charconn
charconn
char
s
char
-
charconn
free
charconn
S
F
deflink
end_page

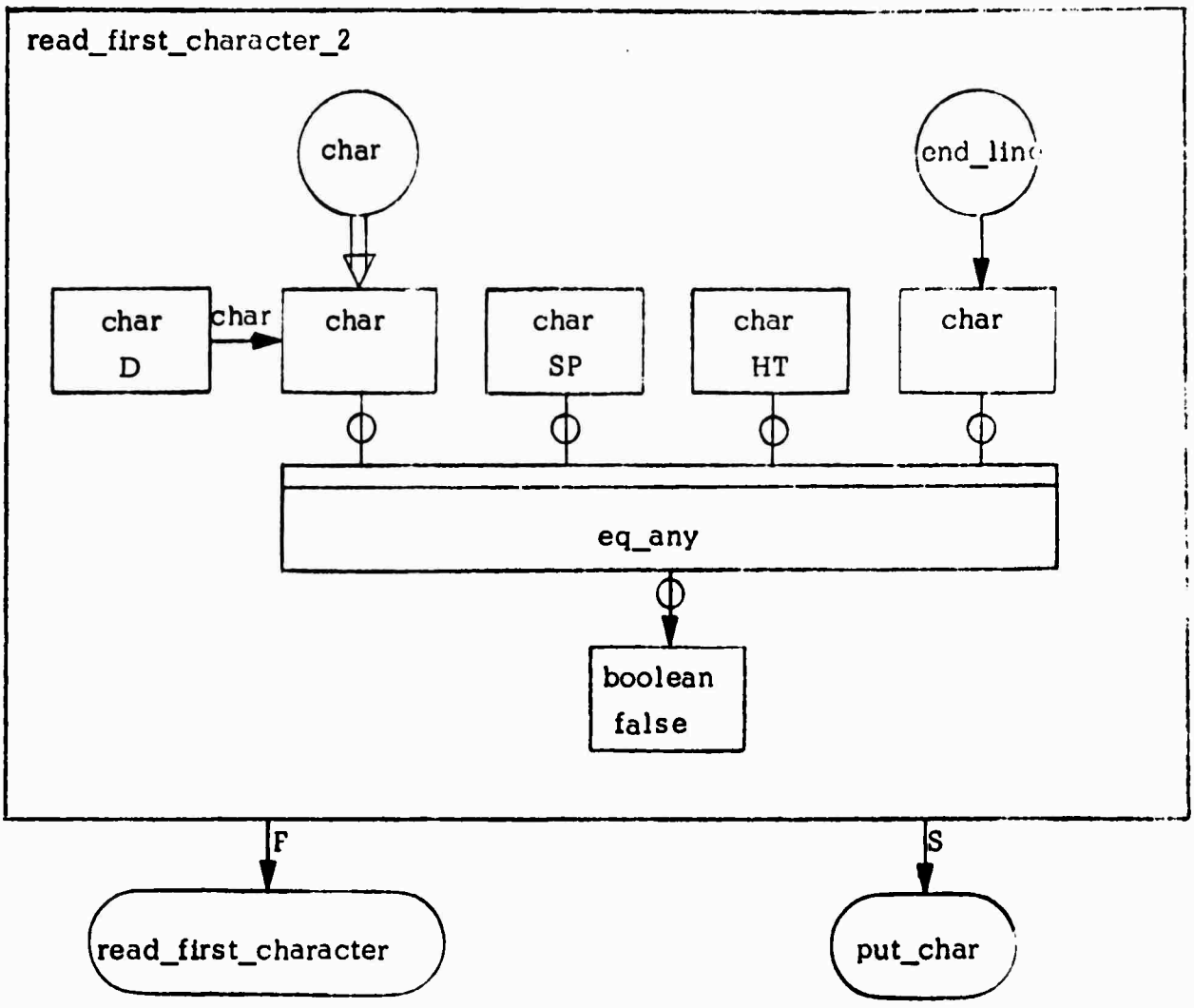

deflink
char
charconn
charconn
special
SP
charconn
free
charconn
SF
deflink_1

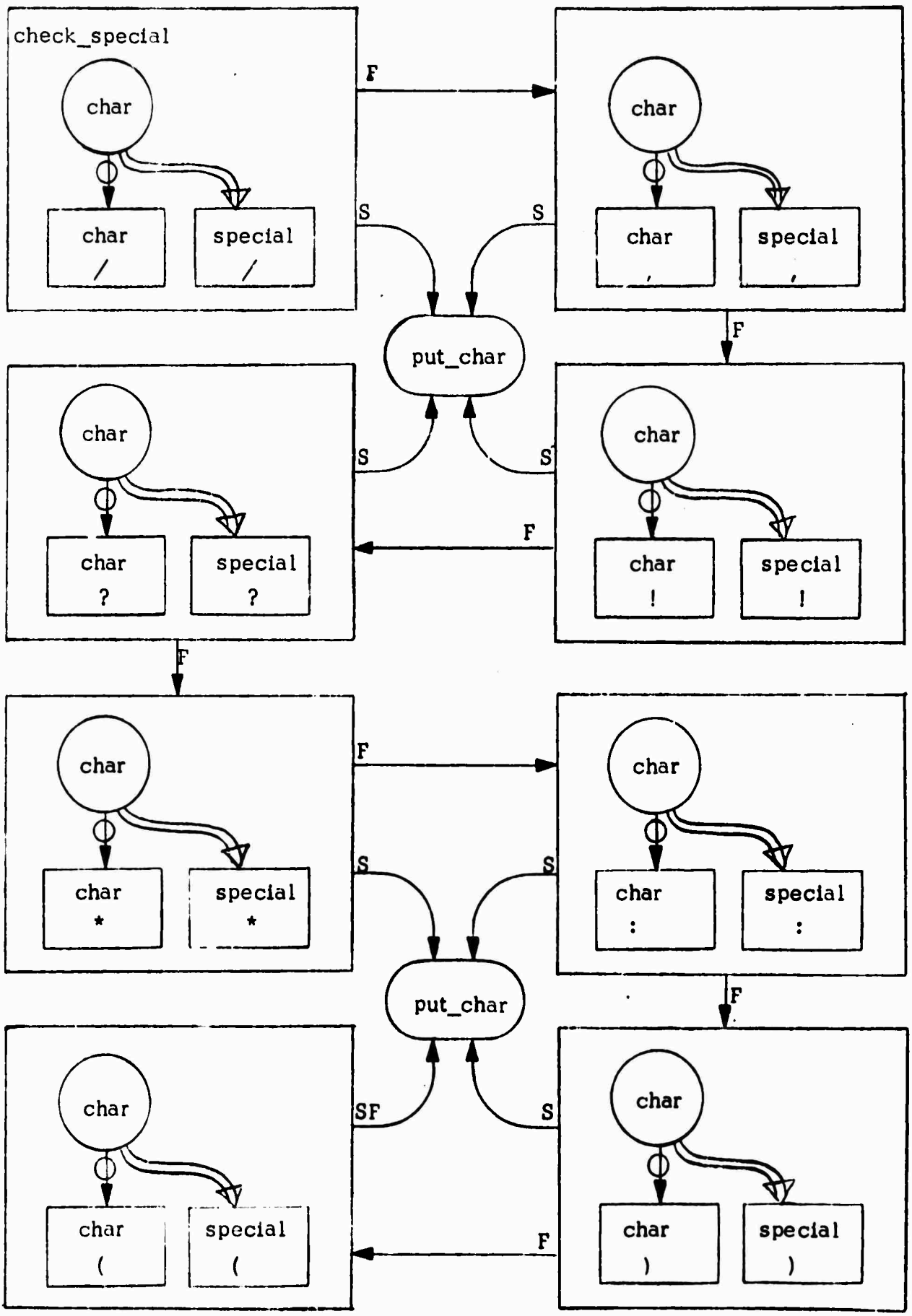

deflink_1
charconn free
charconn
charconn
get_word
charconn
cell h
cell end
type
locate
char
charconn
charconn
special
charconn free
charconn
F
S
deflink_next

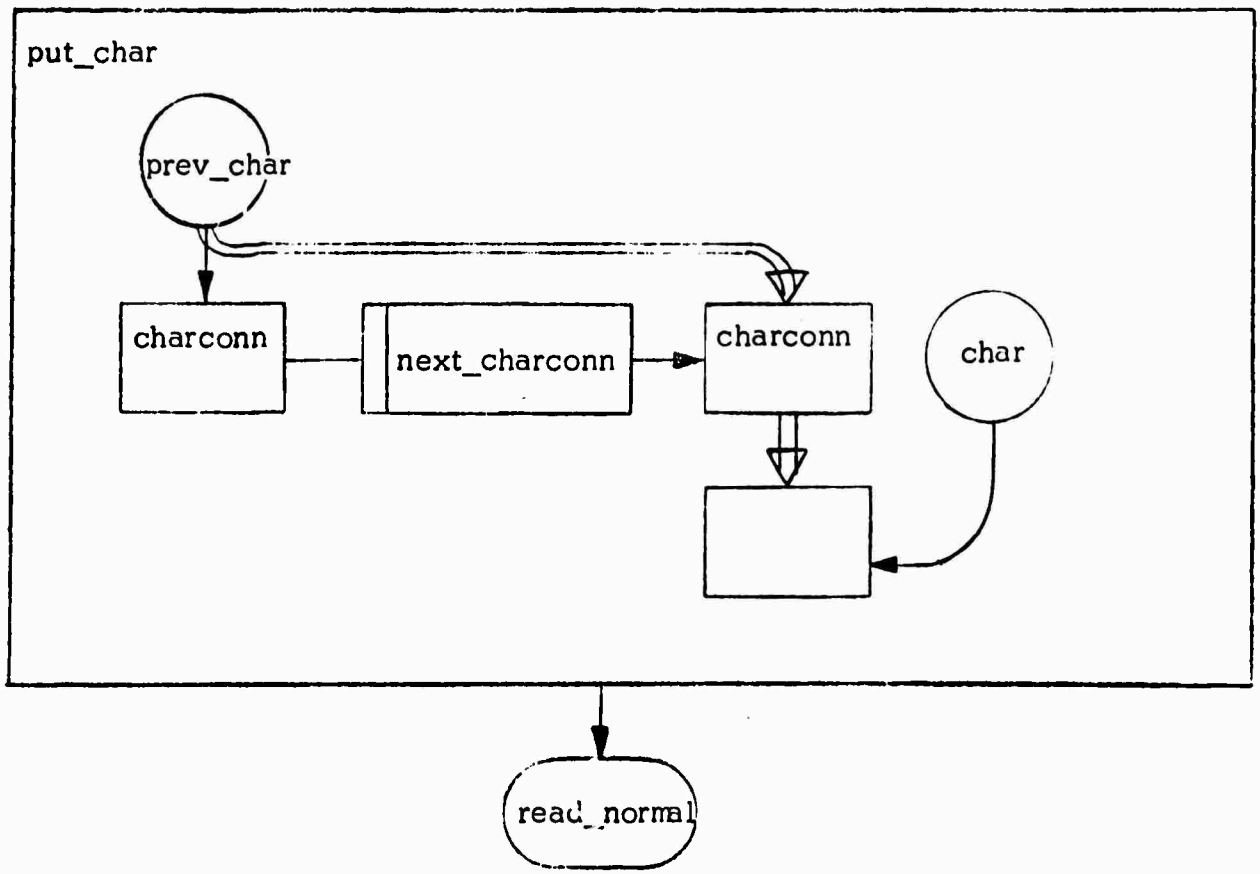

deflink_next
char
charconn
end
S
next_link
F
char
charconn
special
)
S
char
charconn
free_charconn_list
F
S
cell
h
read_function
builtin
link
write_function
get_link_name
special
special
)
F
deflink_next

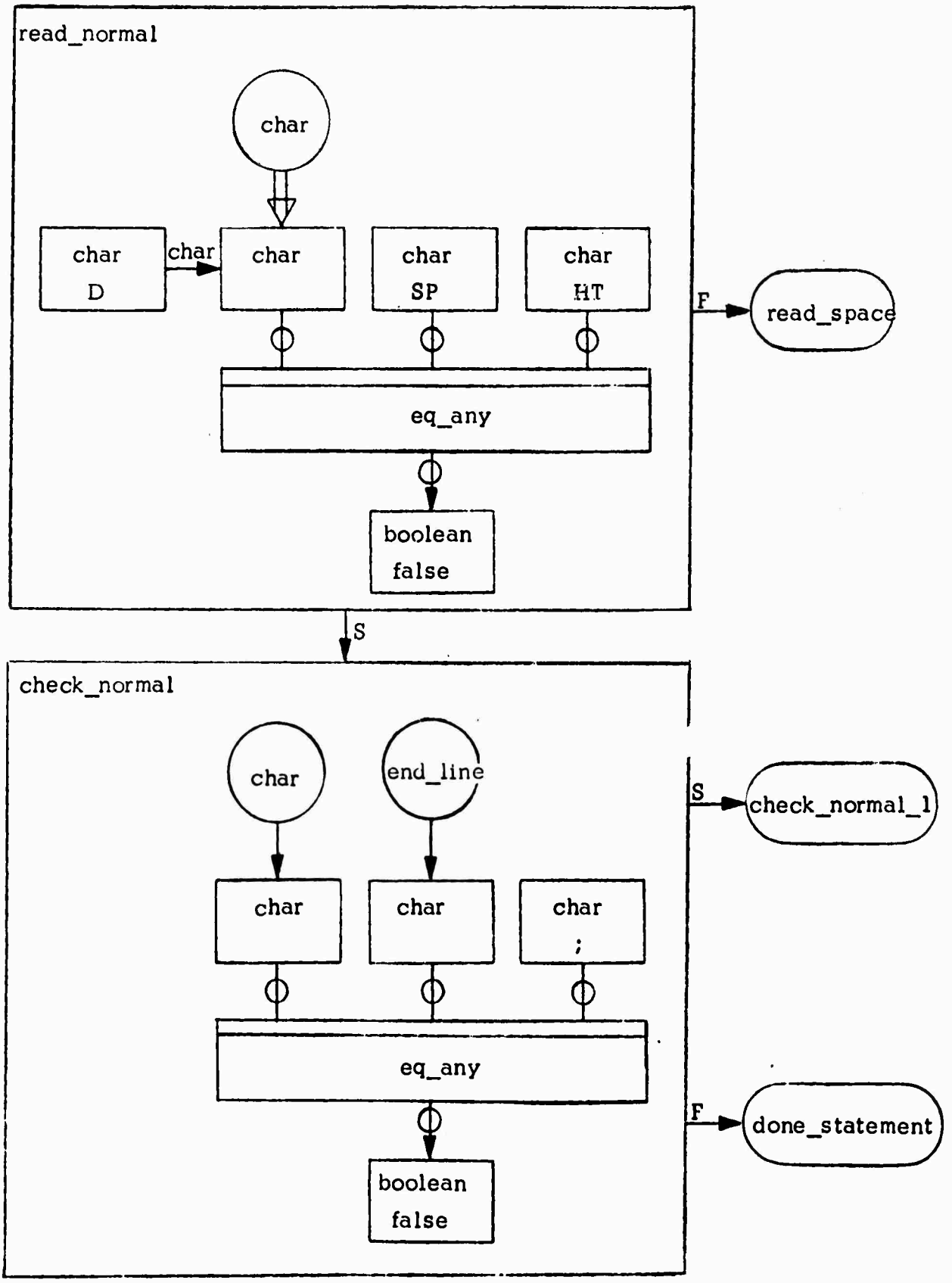

end_page
pname_list
next_pname
pname
free_next_pname
next_pname
pname
pname
charconn
end
S
minus_monitor
F
next_pname
pname
pname
pname
charconn
free_charconn_list

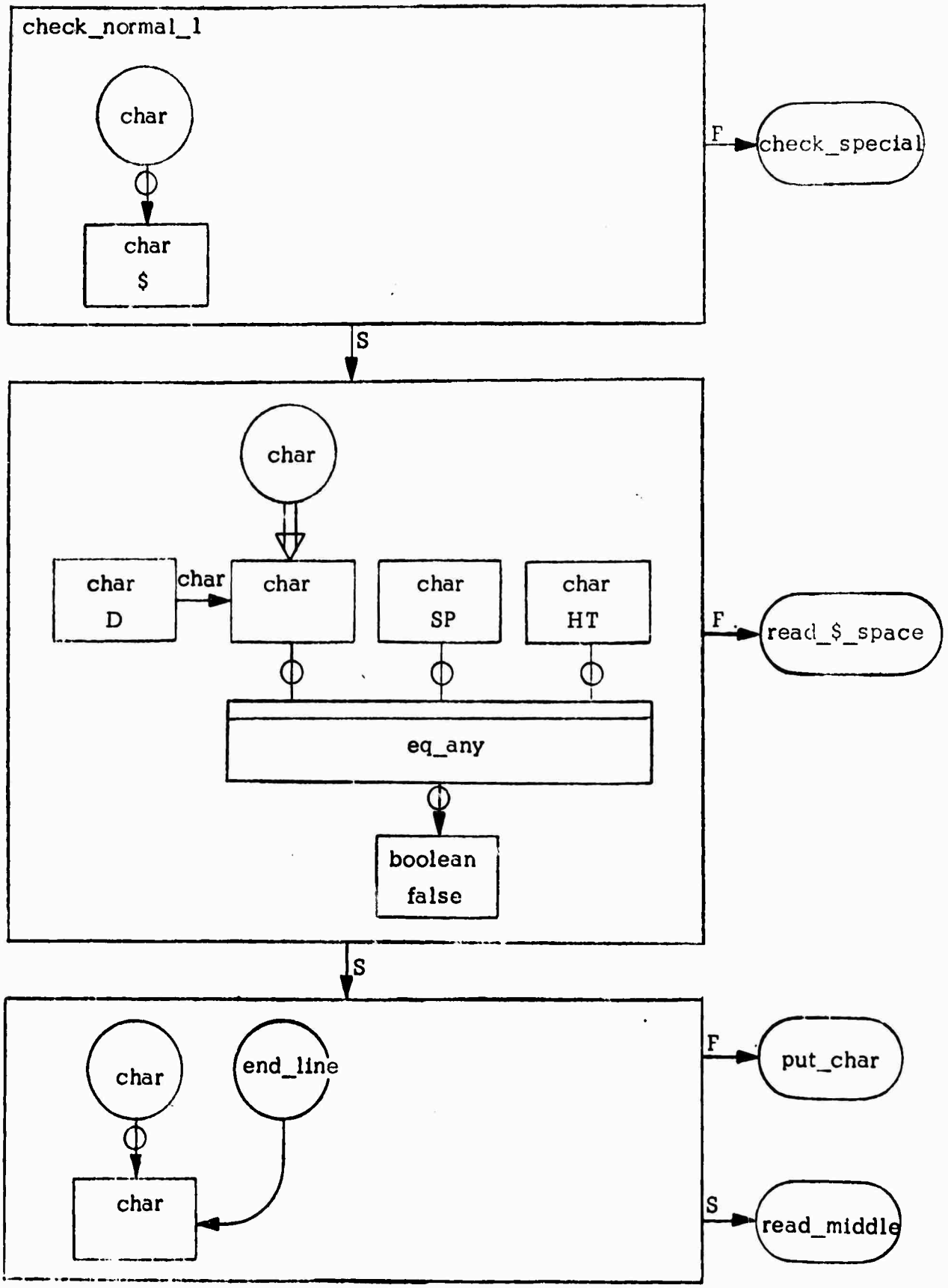

get_statement
prev_char
charconn
free
read_first_character
char
end_line
char
D
char
char
char
SP
char
HT
char
char
;
eq_any
boolean
false
S
F
char
$
F
check_special
S
read_first_character_2
read_first_char

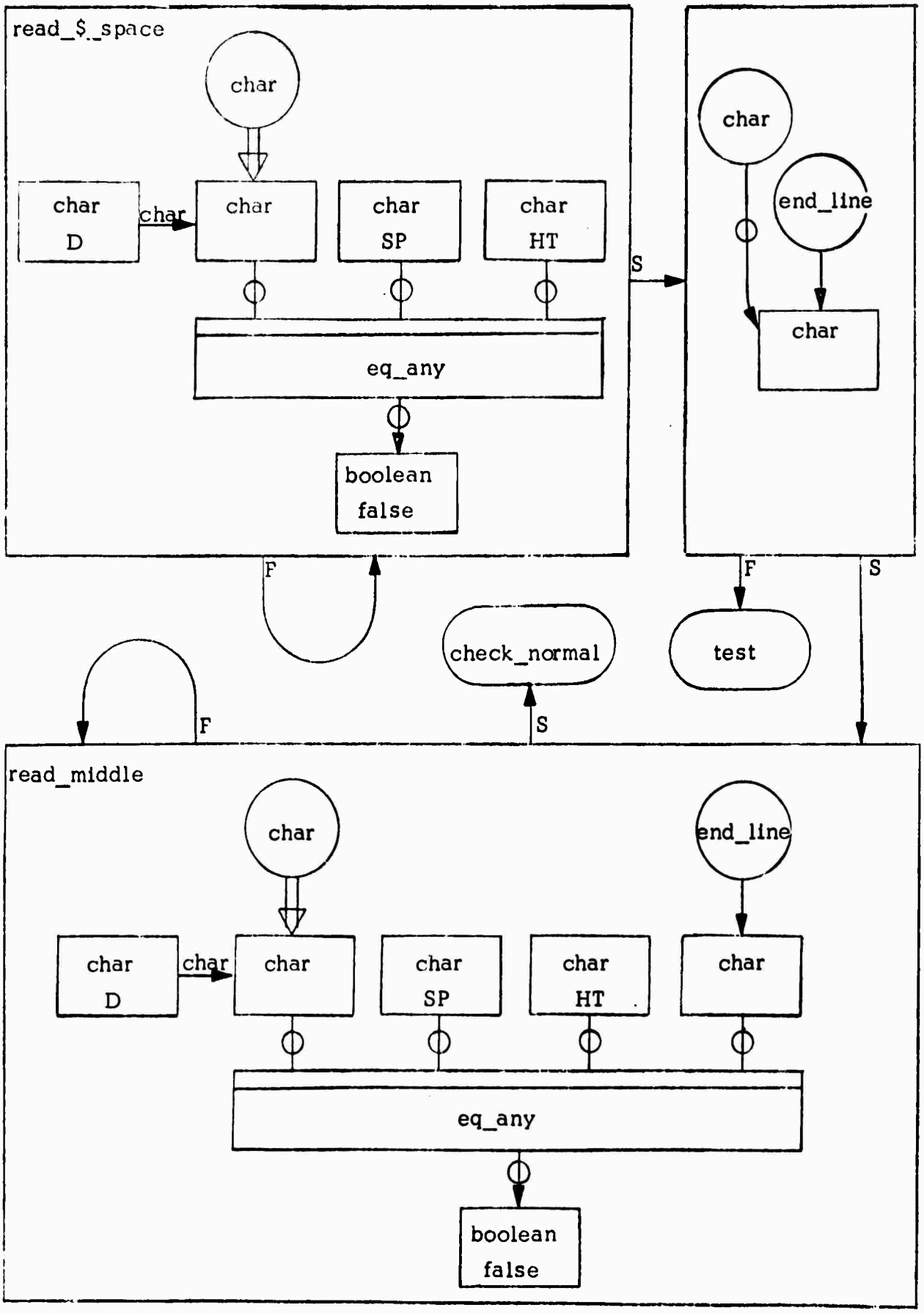

read_first_character_2
char
end_lin
char
D
char
char
char
SP
char
HT
char
eq_any
boolean
false
F
S
read_first_character
put_char

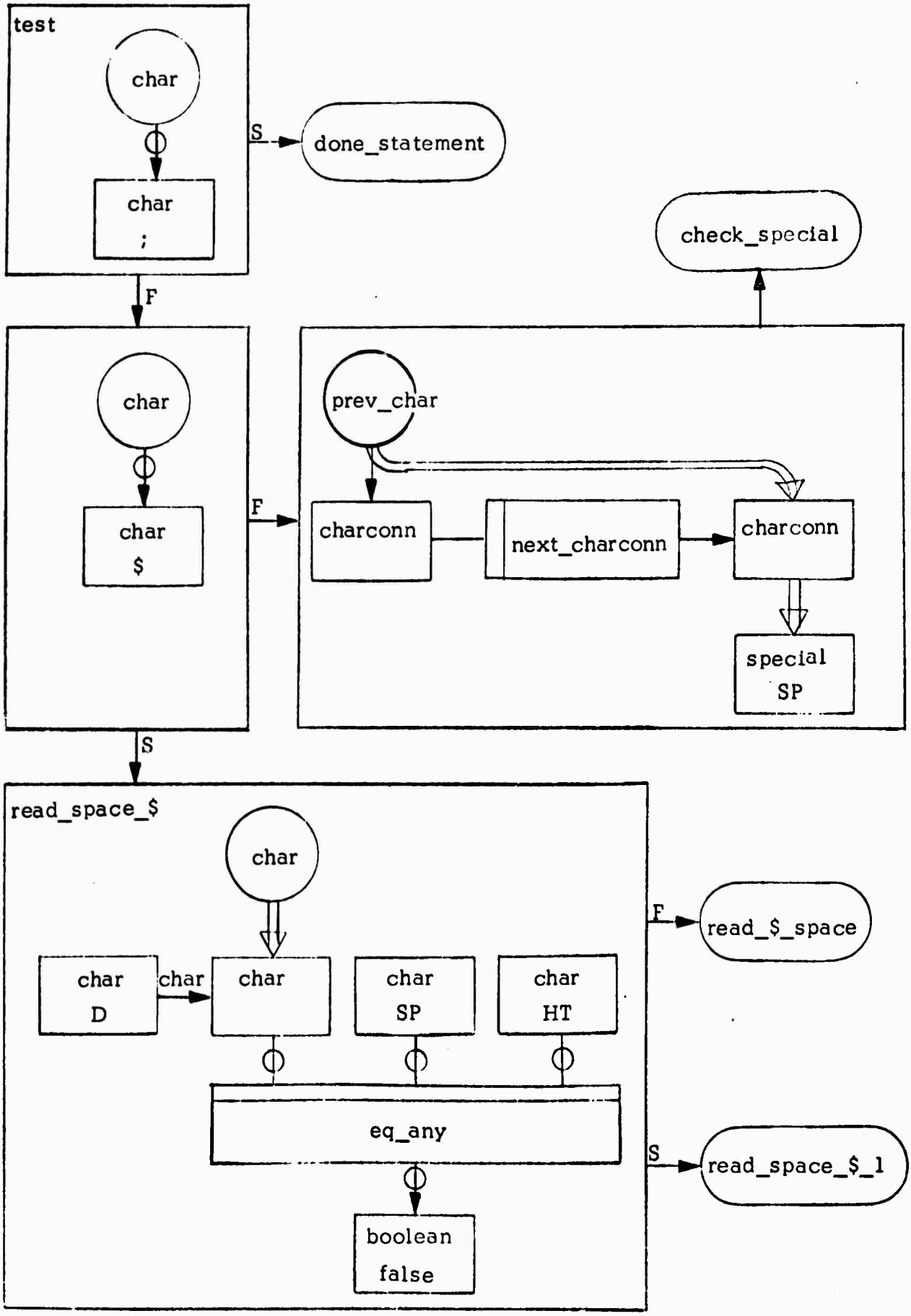

check_special
char
char
/
special
/
F
S
S
char
char
,
special
,
F
put_char
char
char
?
special
?
S
S
F
char
char
!
special
!
F
char
char
*
special
*
F
S
S
char
char
:
special
:
F
put_char
char
char
(
special
(
SF
S
F
char
char
)
special
)

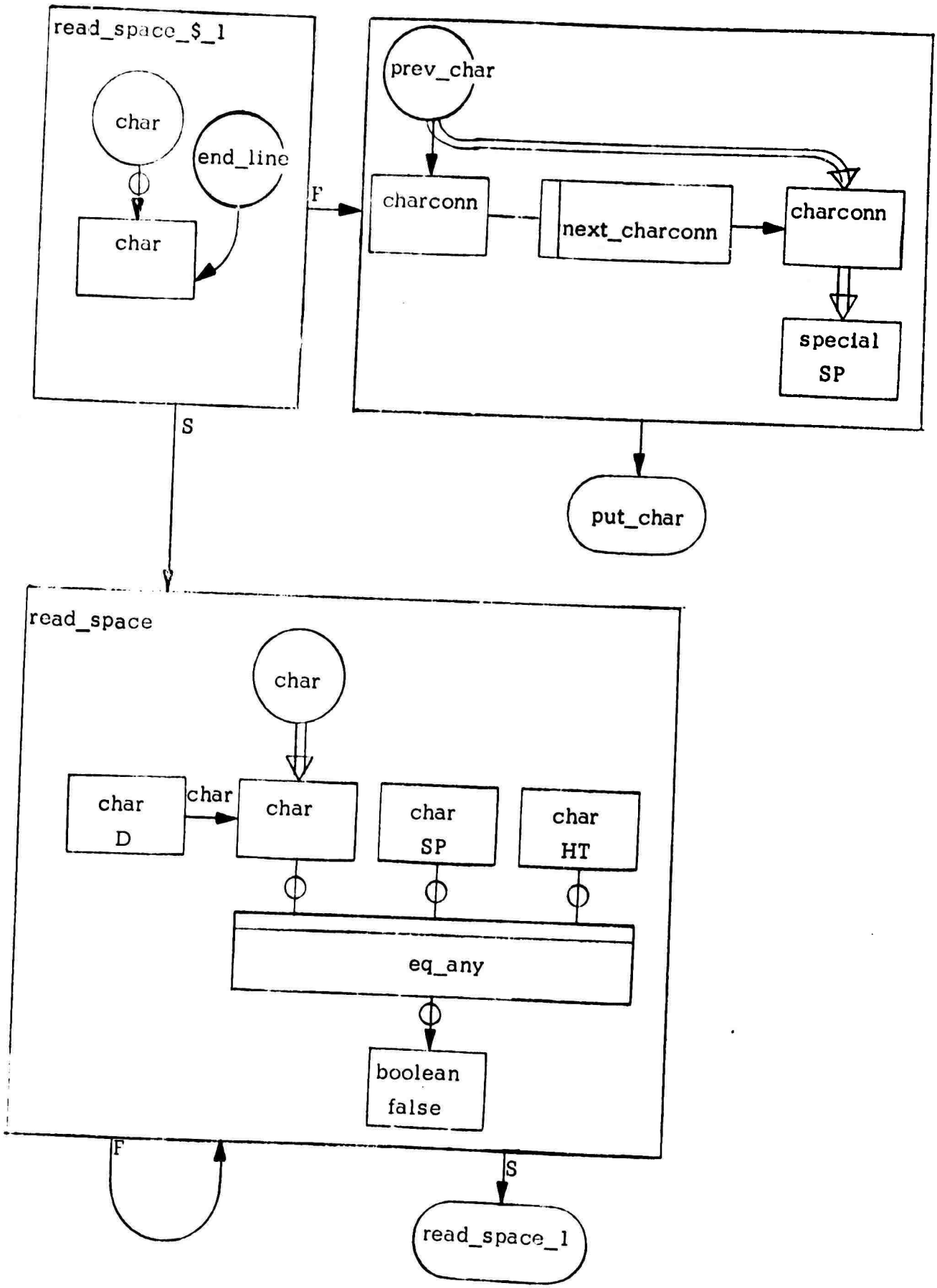

put_char
prev_char
charconn
next_charconn
charconn
char
read_normal

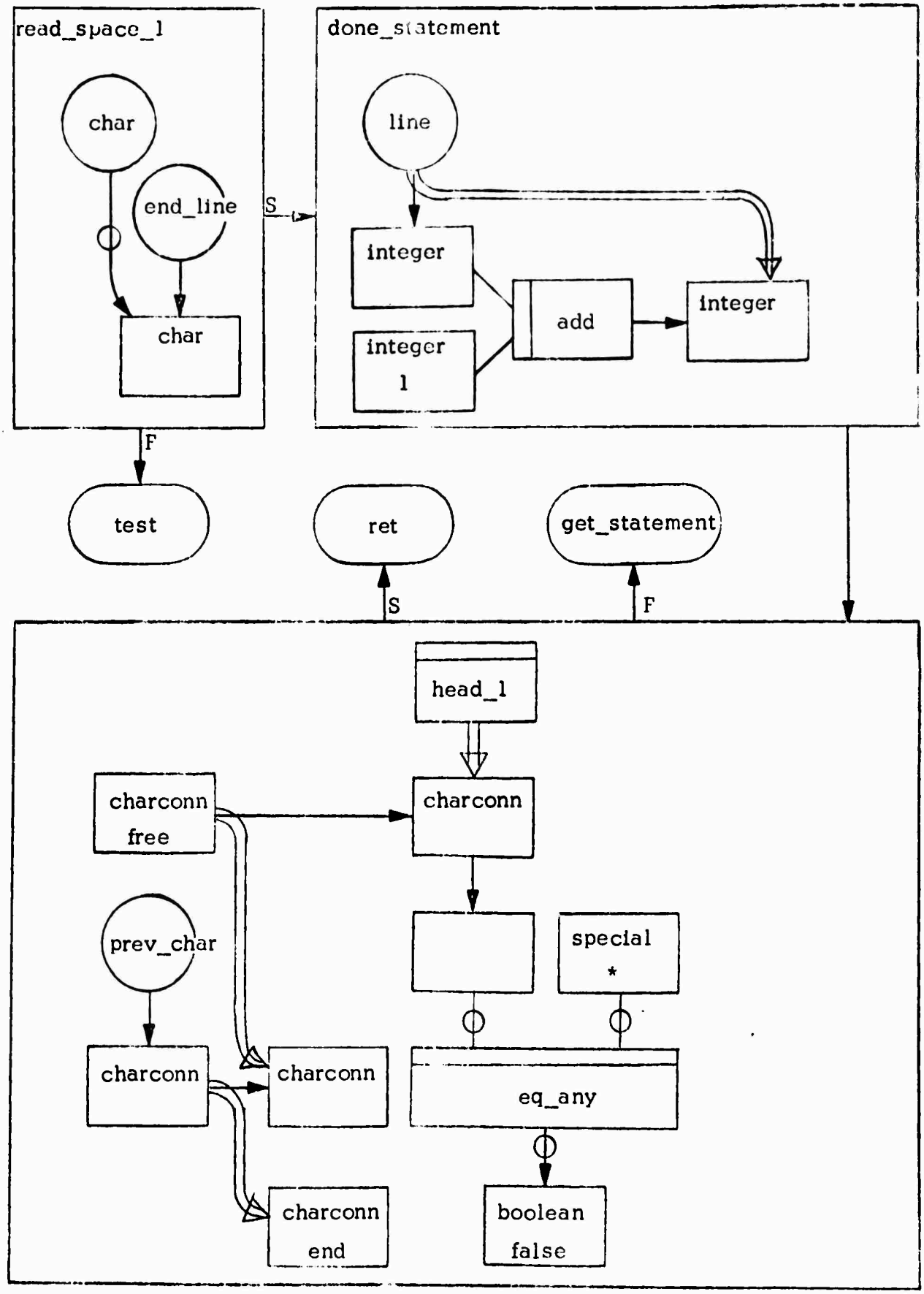

read_normal
char
char
D
char
char
SP
char
HT
eq_any
boolean
false
F
read_space
S
check_normal
char
end_line
char
char
char
;
eq_any
boolean
false
S
check_normal_1
F
done_statement

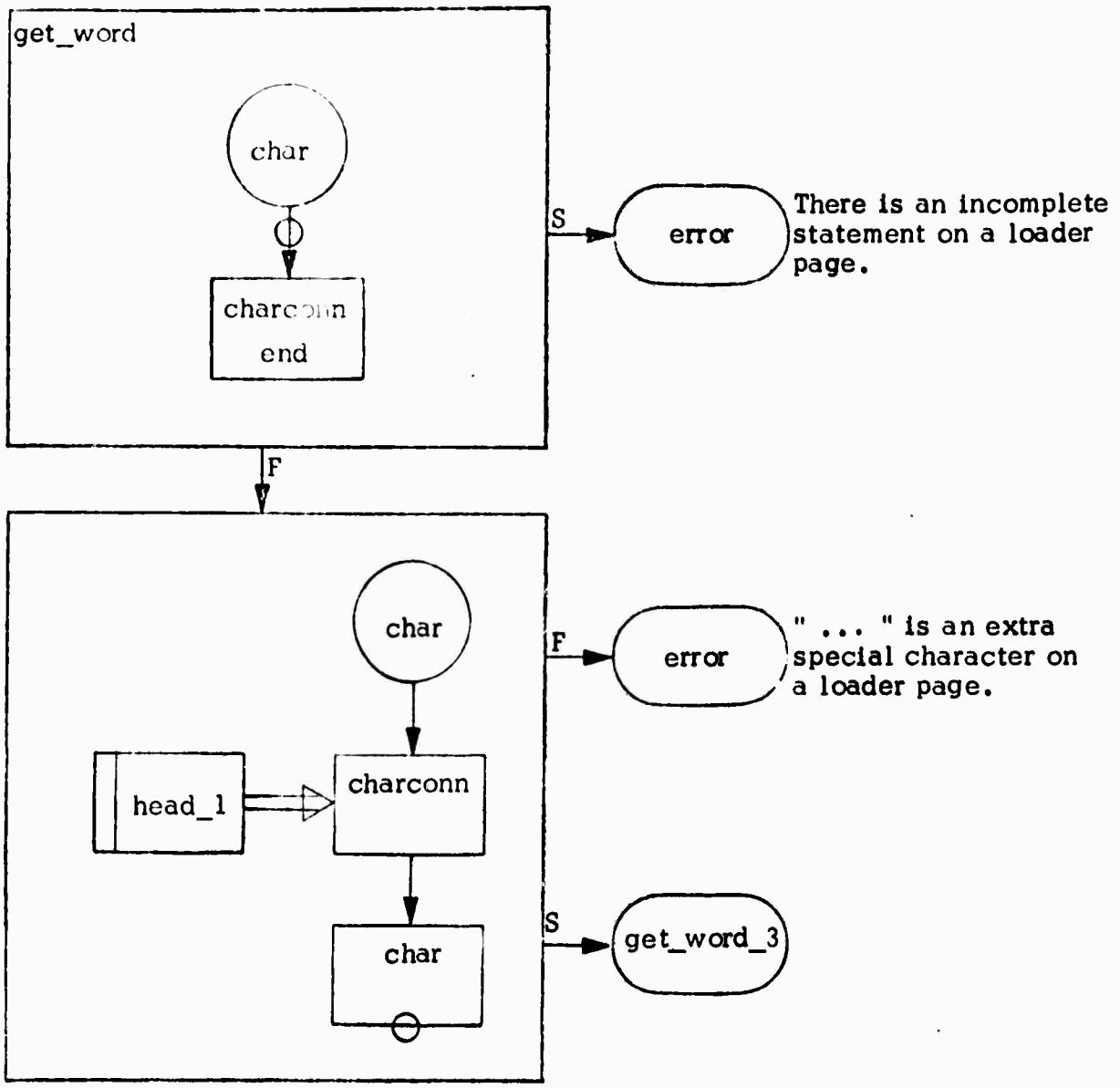

check_normal_1
char
char
$
F
check_special
S
char
char
D
char
char
char
SP
char
HT
F
read_$_space
eq_any
boolean
false
S
char
end_line
char
F
put_char
S
read_middle

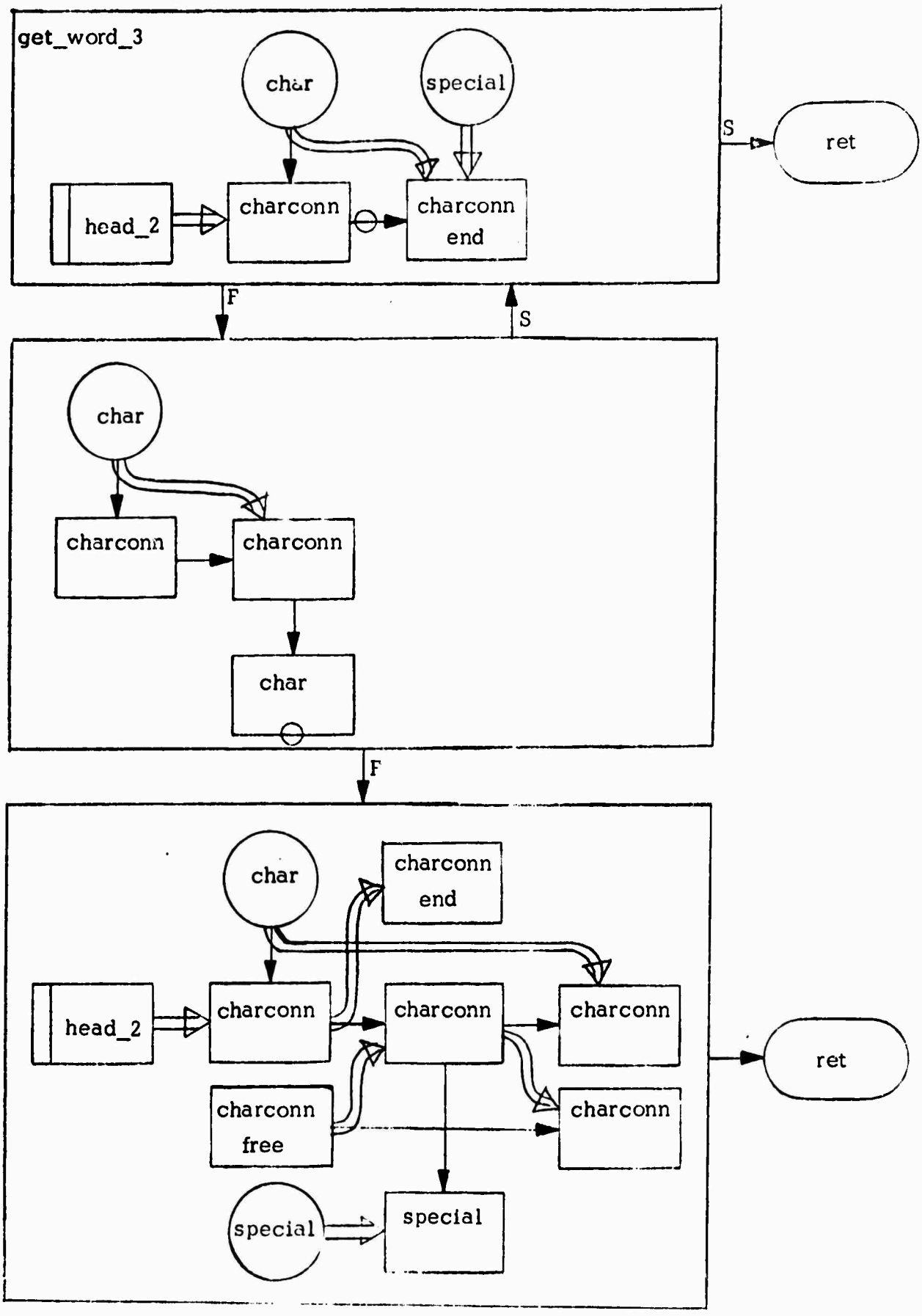

read_$_space
char
char
D
char
char
char
SP
char
HT
eq_any
boolean
false
S
char
end_line
char
F
S
F
check_normal
test
F
S
read_middle
char
end_line
char
D
char
char
char
SP
char
HT
char
eq_any
boolean
false

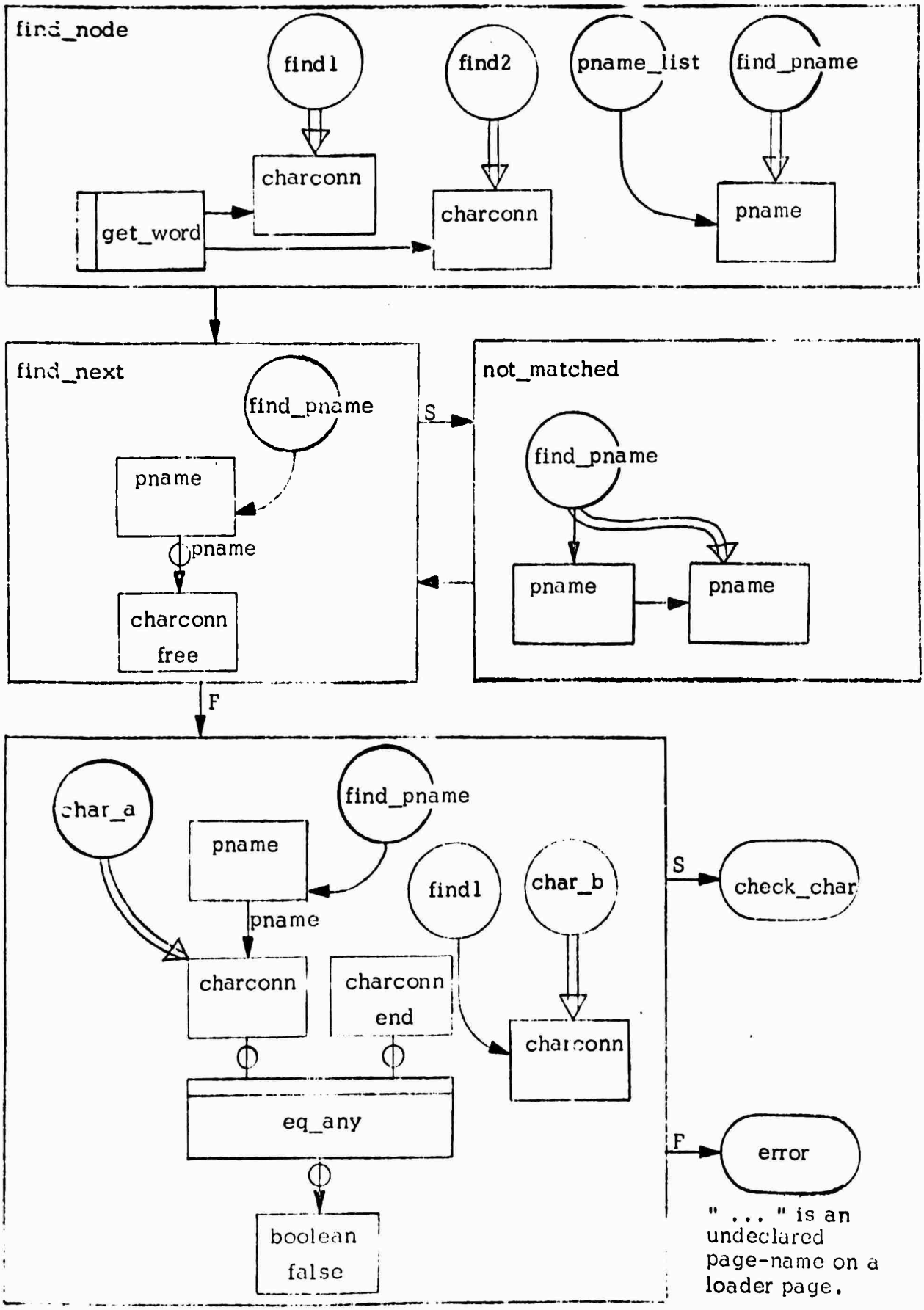

test
char
char
;
S
done_statement
F
check_special
char
char
$
F
prev_char
charconn
next_charconn
charconn
special
SP
S
read_space_$
char
char
D
char
char
char
SP
char
HT
F
read_$_space
eq_any
S
read_space_$_1
boolean
false

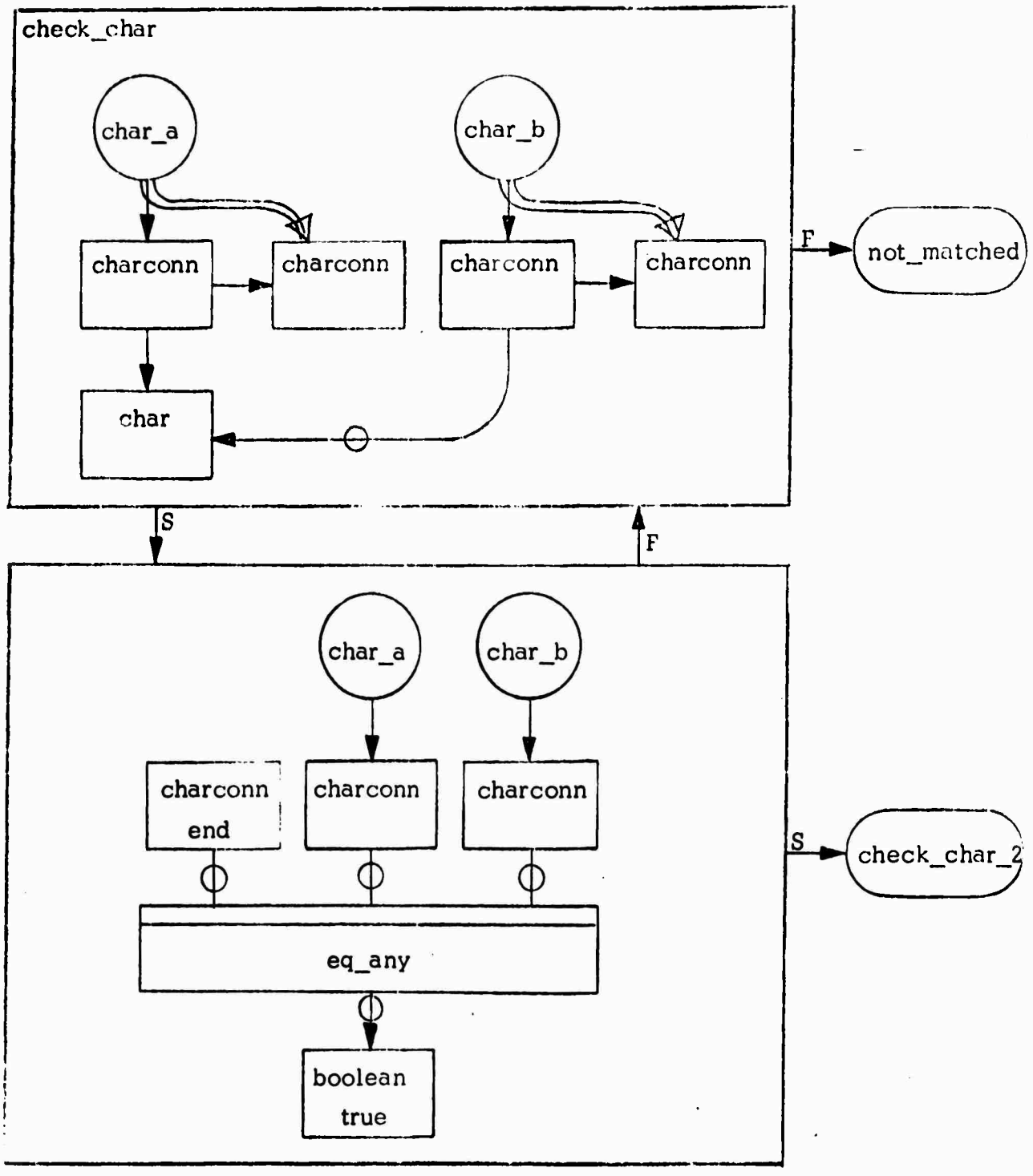

read_space_$_1
char
end_line
char
F
prev_char
charconn
next_charconn
charconn
special
SP
S
put_char
read_space
char
char
D
char
char
SP
char
HT
eq_any
boolean
false
F
S
read_space_1

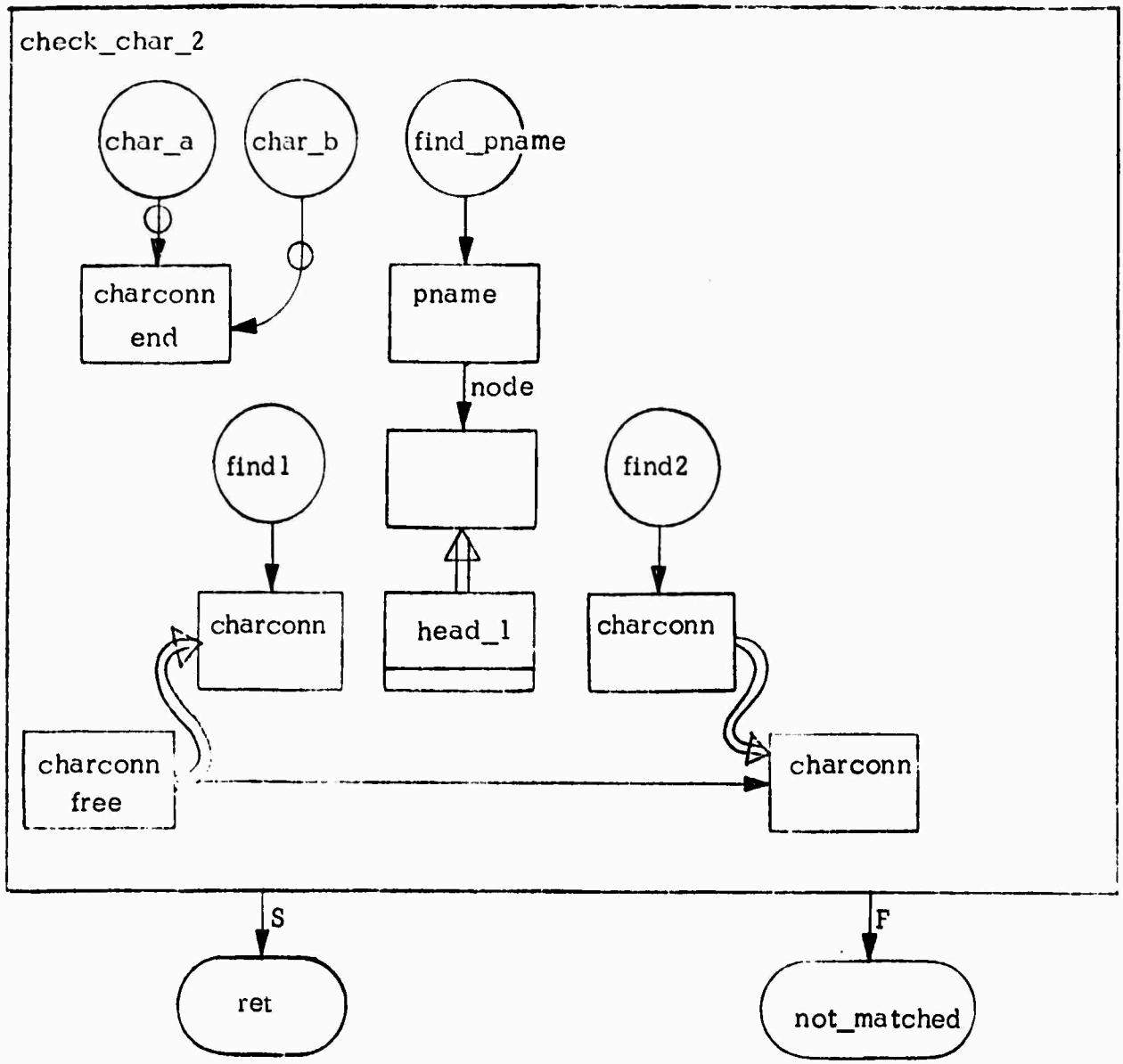

read_space_1
char
end_line
char
S
done_statement
line
integer
integer
1
add
integer
F
test
ret
get_statement
S
F
head_1
charconn
free
charconn
prev_char
special
*
charconn
charconn
eq_any
charconn
end
boolean
false

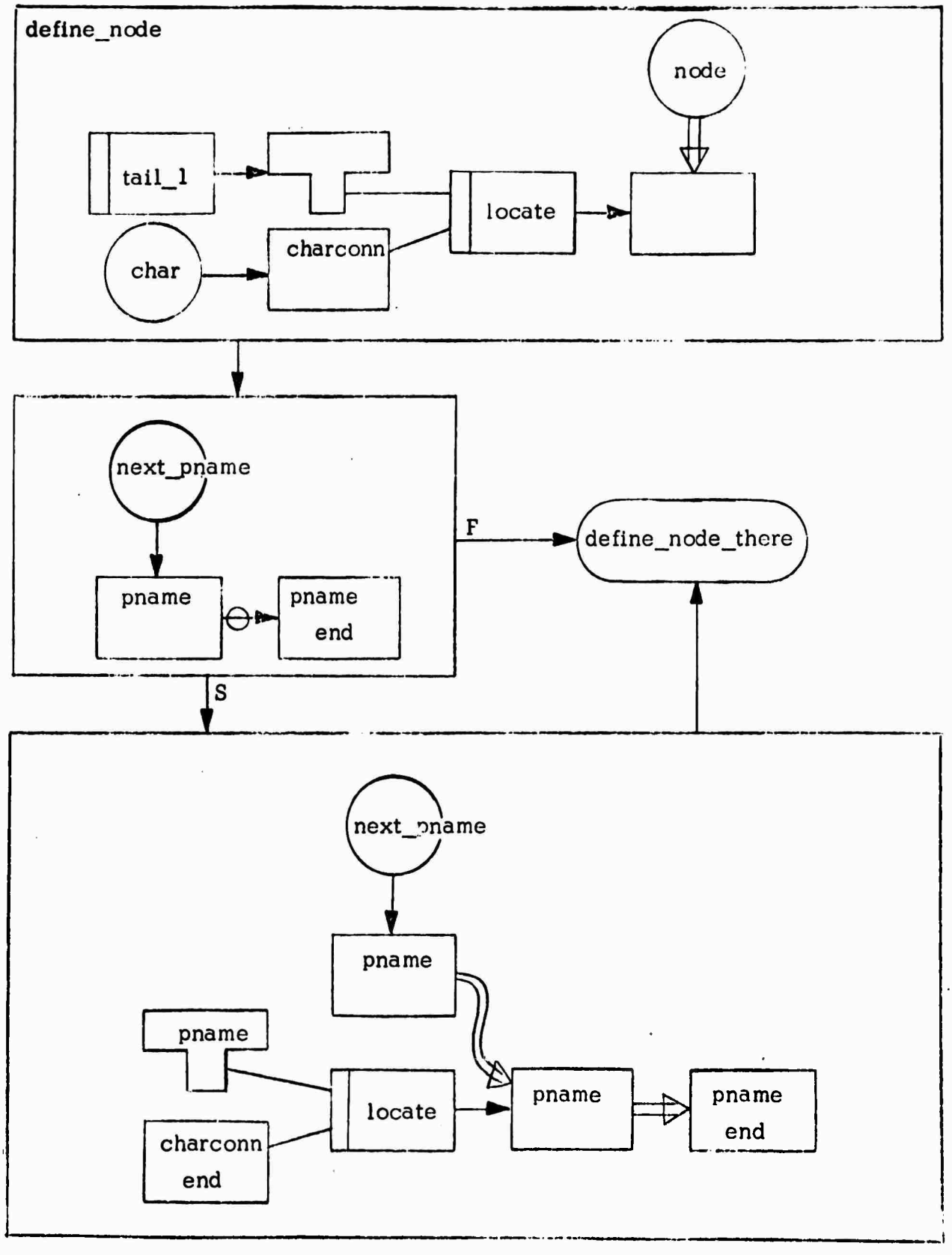

get_word
char
charconn
end
S
error
There is an incomplete statement on a loader page.
F
char
F
error
" ... " is an extra special character on a loader page.
head_1
charconn
char
S
get_word_3

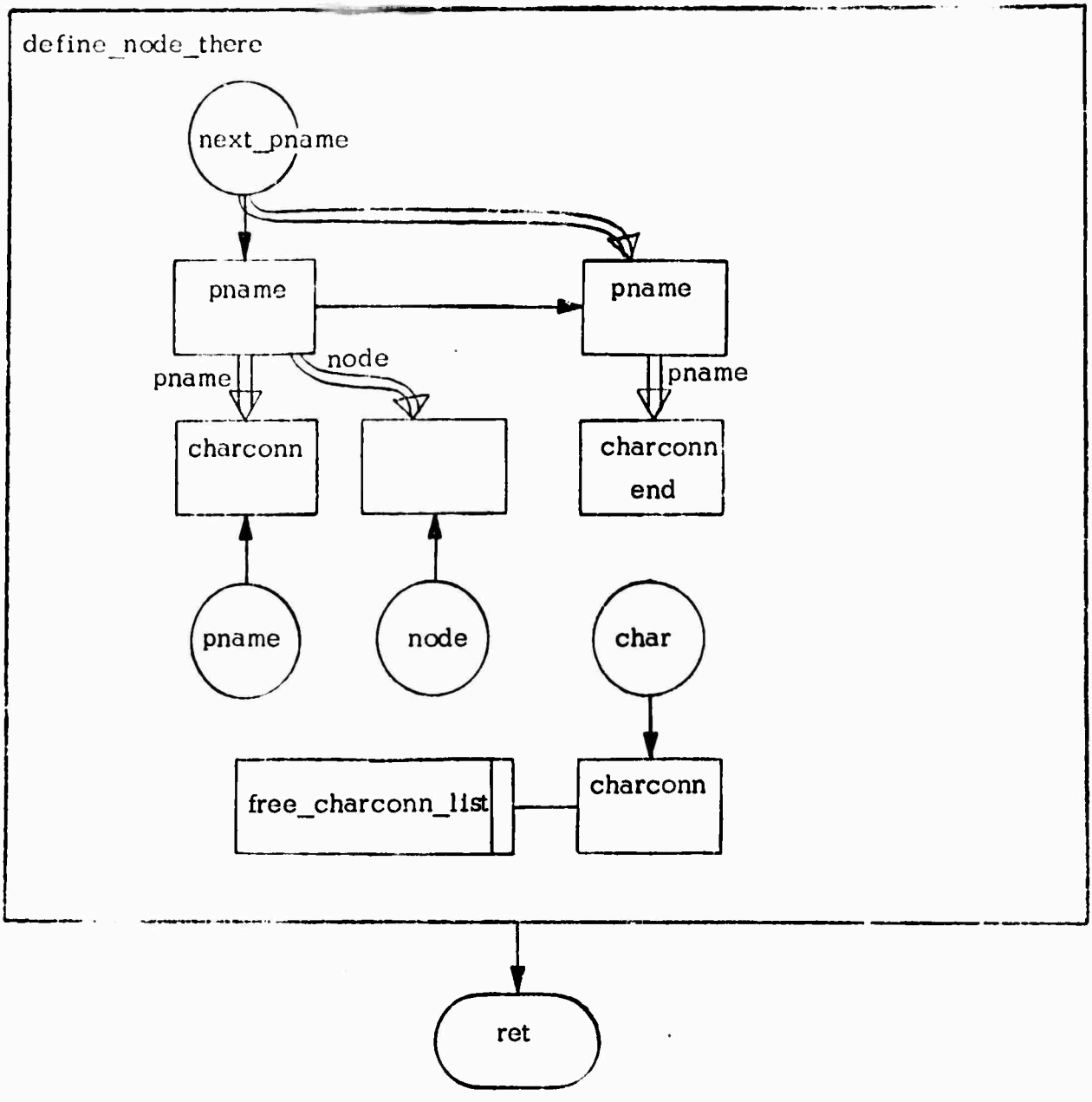

get_word_3
char
special
S
ret
head_2
charconn
charconn
end
F
S
char
charconn
charconn
char
F
char
charconn
end
head_2
charconn
charconn
charconn
charconn
free
charconn
special
special
ret

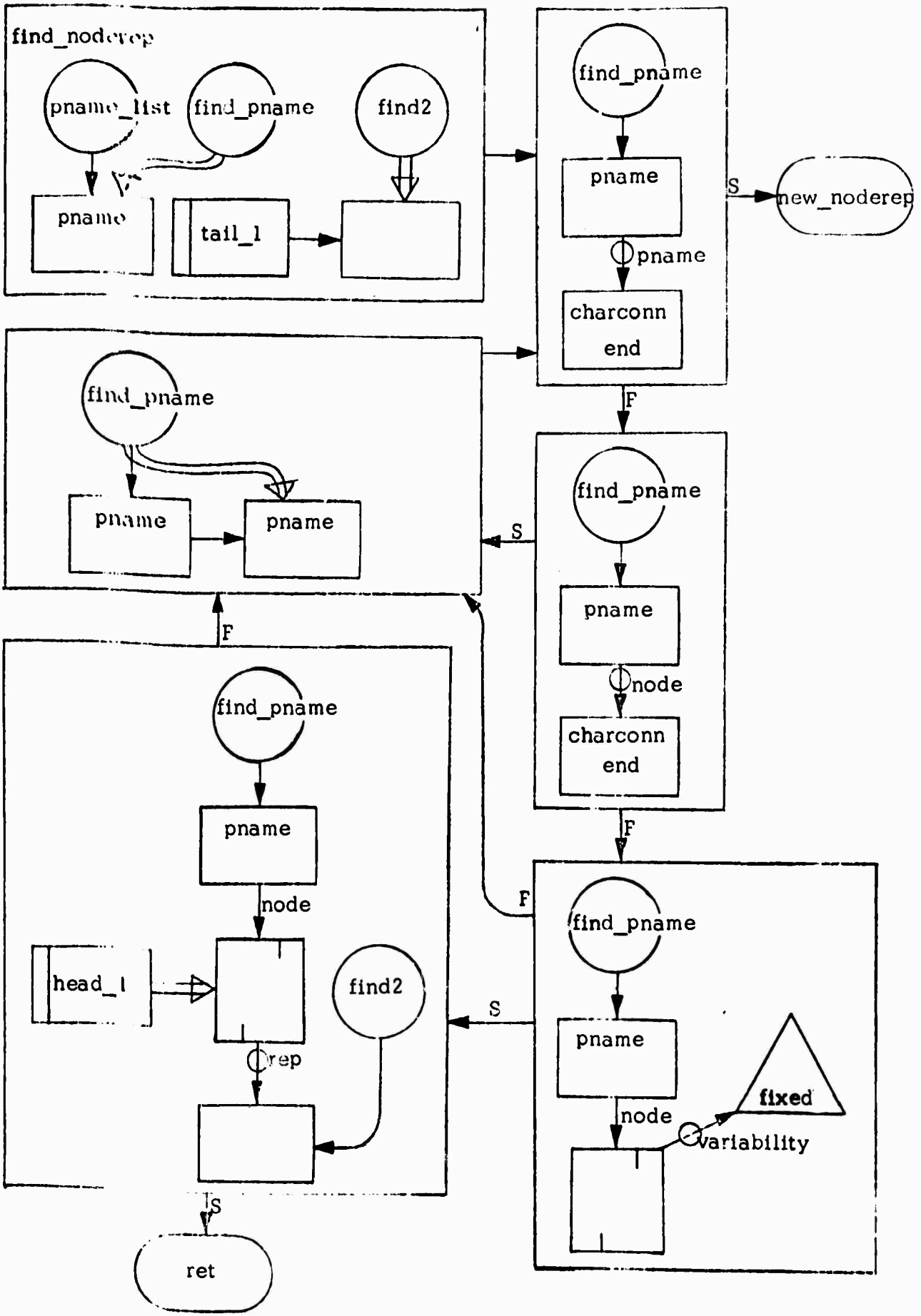

find_node
find1
find2
pname_list
find_pname
charconn
get_word
charconn
pname
find_next
find_pname
pname
pname
charconn
free
S
not_matched
find_pname
pname
pname
F
char_a
find_pname
pname
pname
find1
char_b
charconn
charconn
end
charconn
eq_any
boolean
false
S
check_char
F
error
" ... " is an undeclared page-name on a loader page.

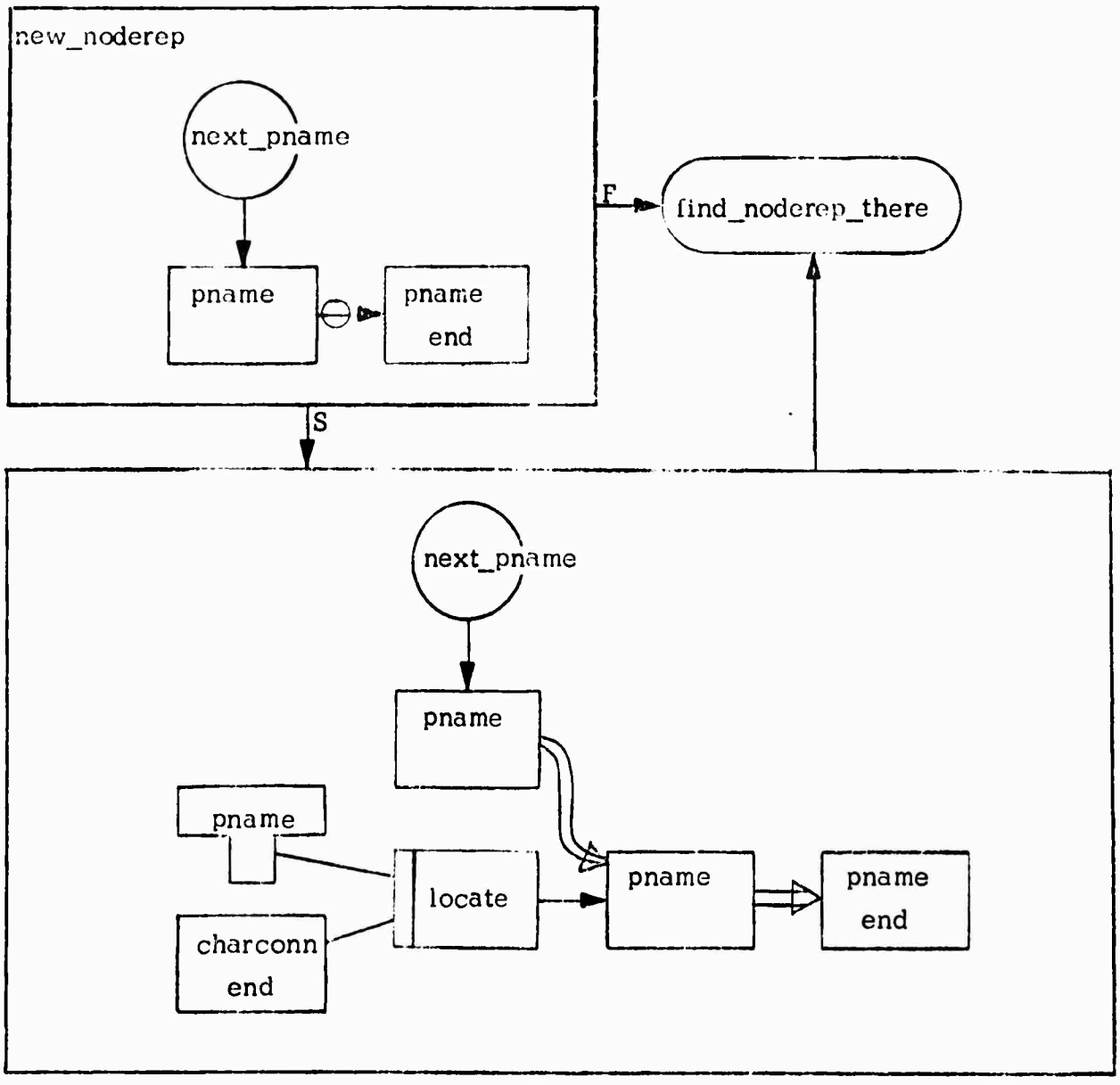

check_char
char_a
char_b
charconn
charconn
charconn
charconn
char
F
not_matched
S
F
char_a
char_b
charconn
end
charconn
charconn
eq_any
boolean
true
S
check_char_2

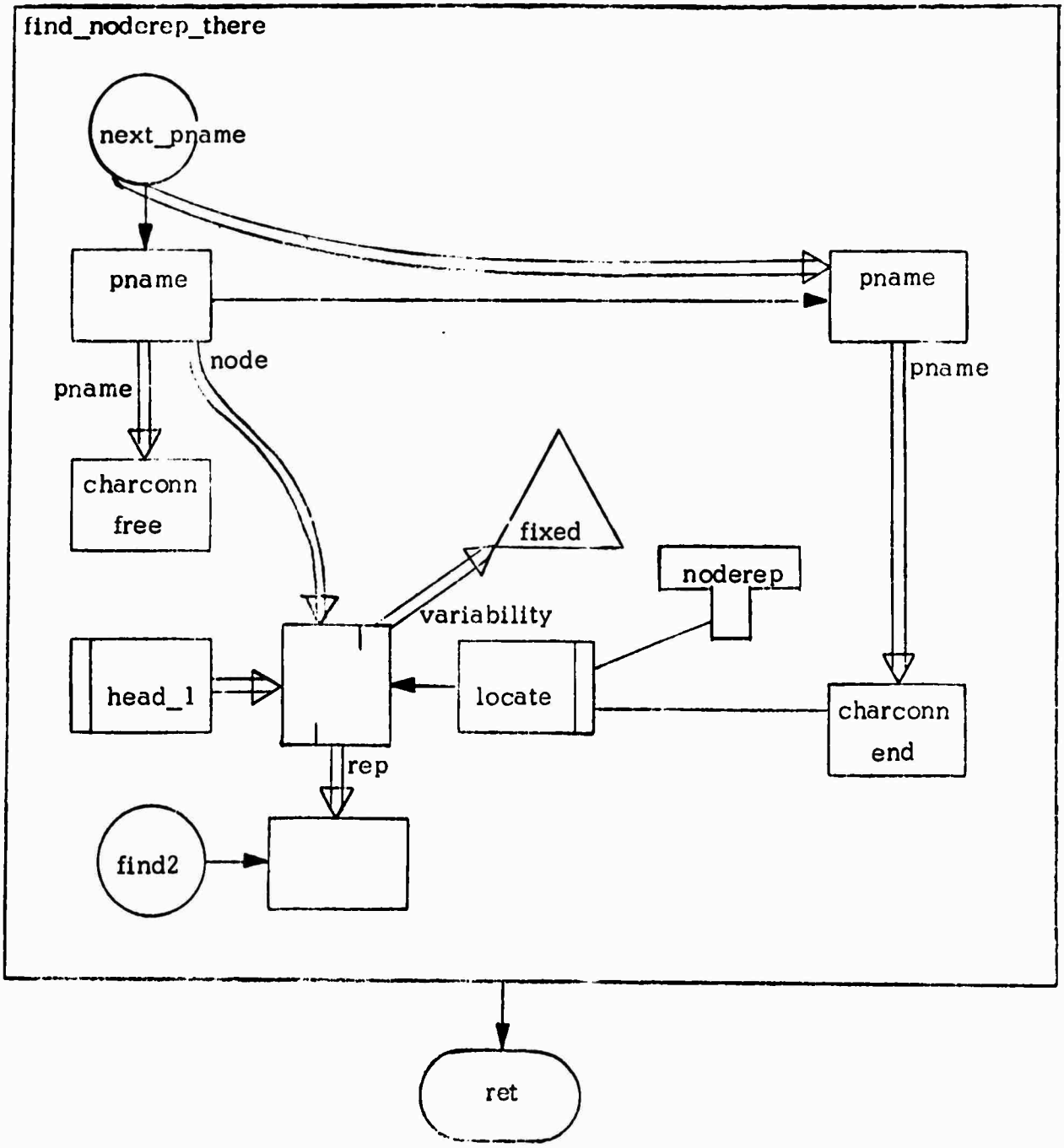

check_char_2
char_a
char_b
find_pname
charconn
end
pname
node
find1
find2
charconn
head_1
charconn
charconn
free
charconn
S
F
ret
not_matched

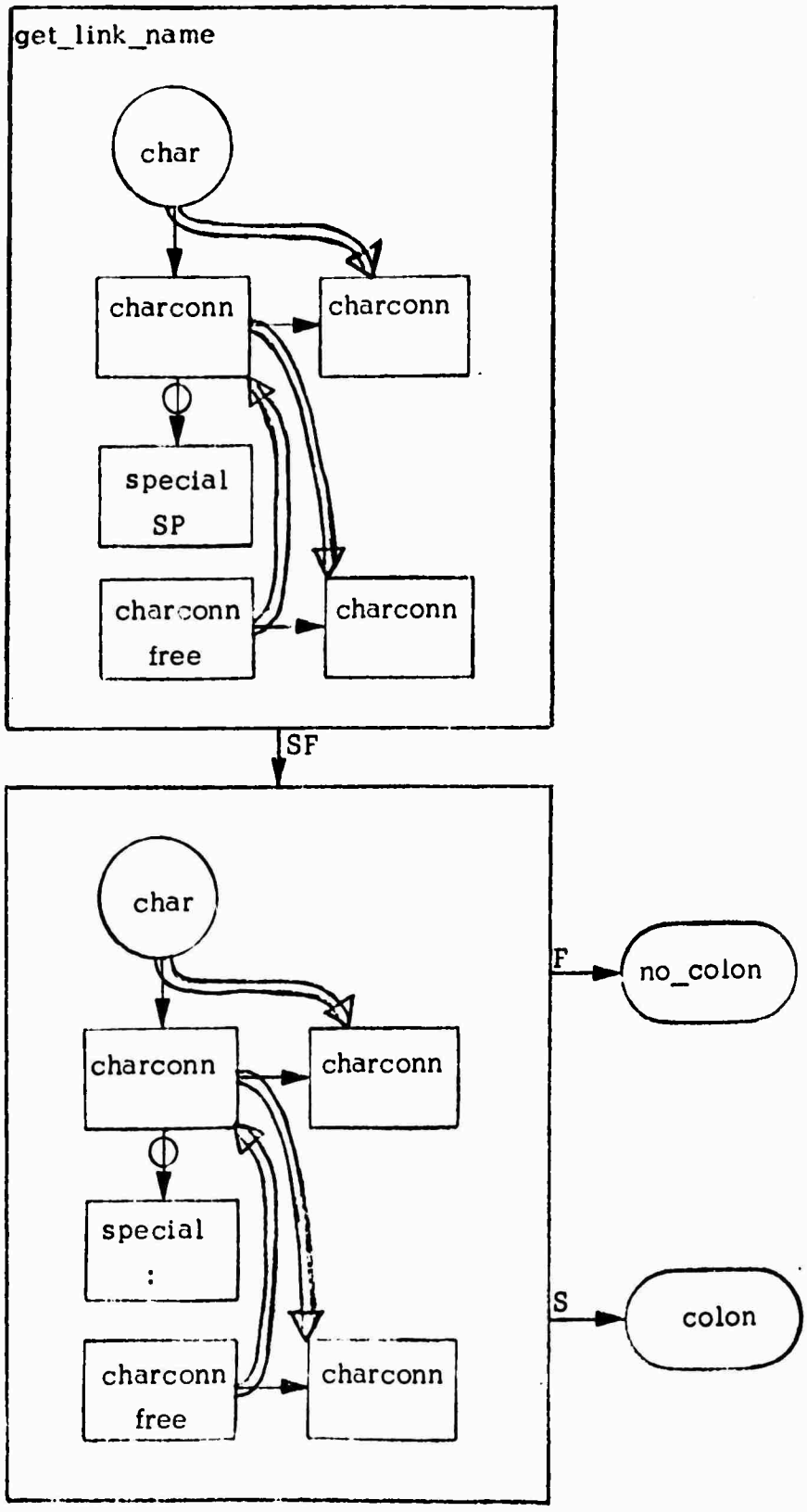

define_node
node
tail_1
locate
char
charconn
next_pname
pname
pname
end
F
define_node_there
S
next_pname
pname
pname
locate
pname
pname
end
charconn
end

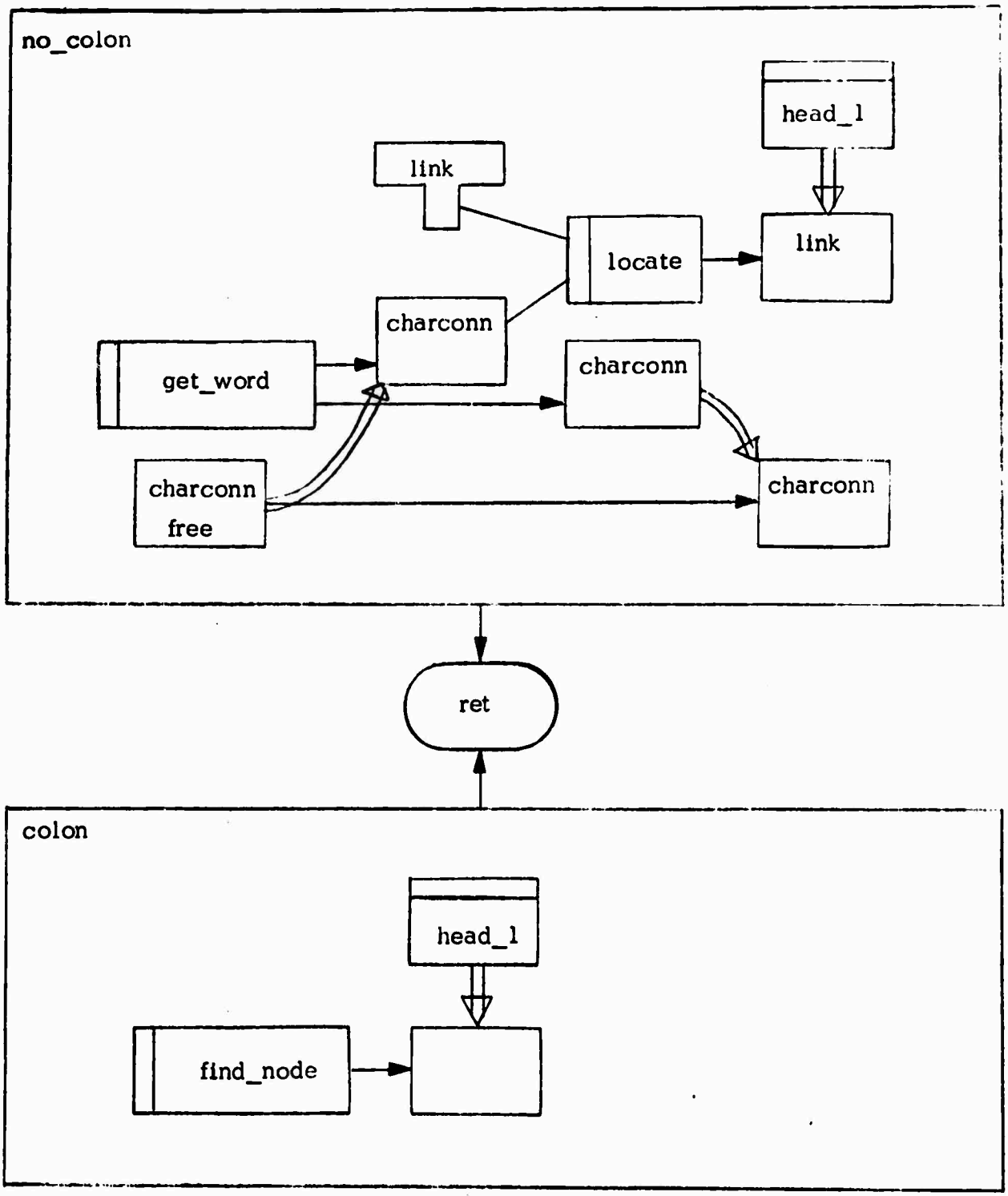

define_node_there
next_pname
pname
pname
pname
node
pname
charconn
charconn
end
pname
node
char
free_charconn_list
charconn
ret

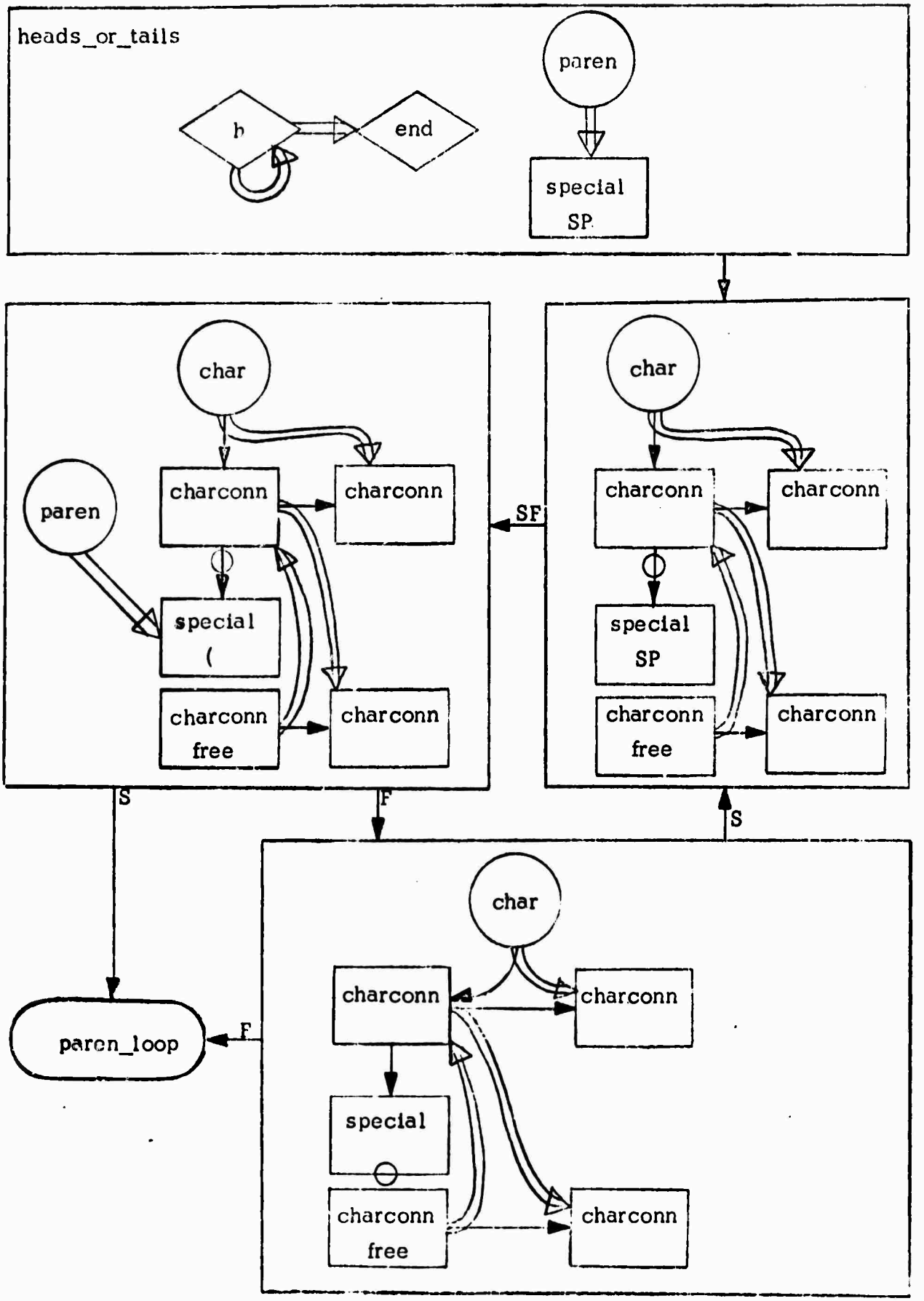

find_noderep
pname_list
find_pname
find2
pname
tail_1
find_pname
pname
pname
charconn
end
S
new_noderep
F
find_pname
pname
pname
S
find_pname
pname
node
charconn
end
F
F
find_pname
pname
node
head_1
find2
rep
S
find_pname
pname
node
fixed
variability
S
ret

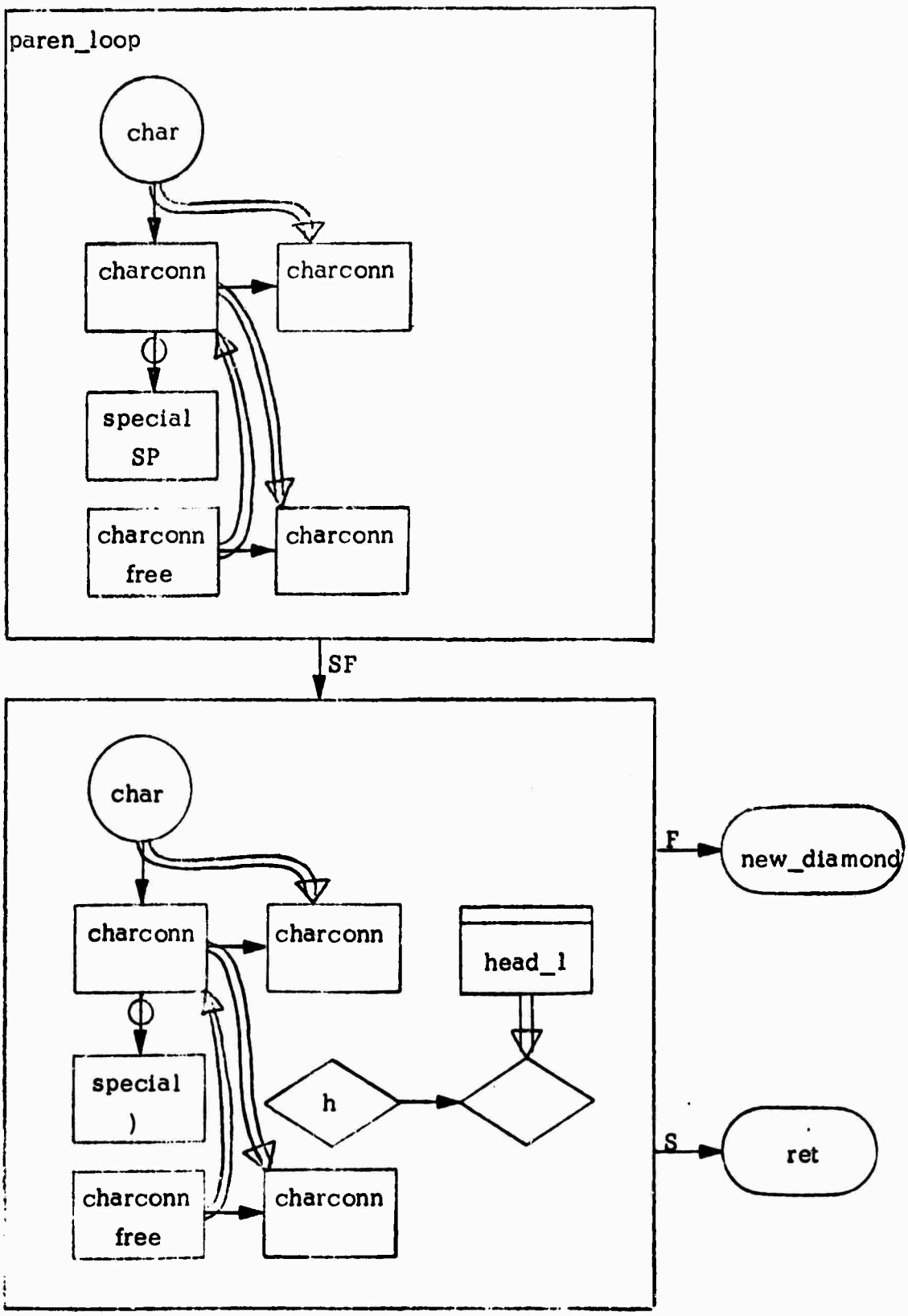

new_noderep
next_pname
pname
pname
end
F
find_noderep_there
S
next_pname
pname
pname
locate
charconn
end
pname
pname
end

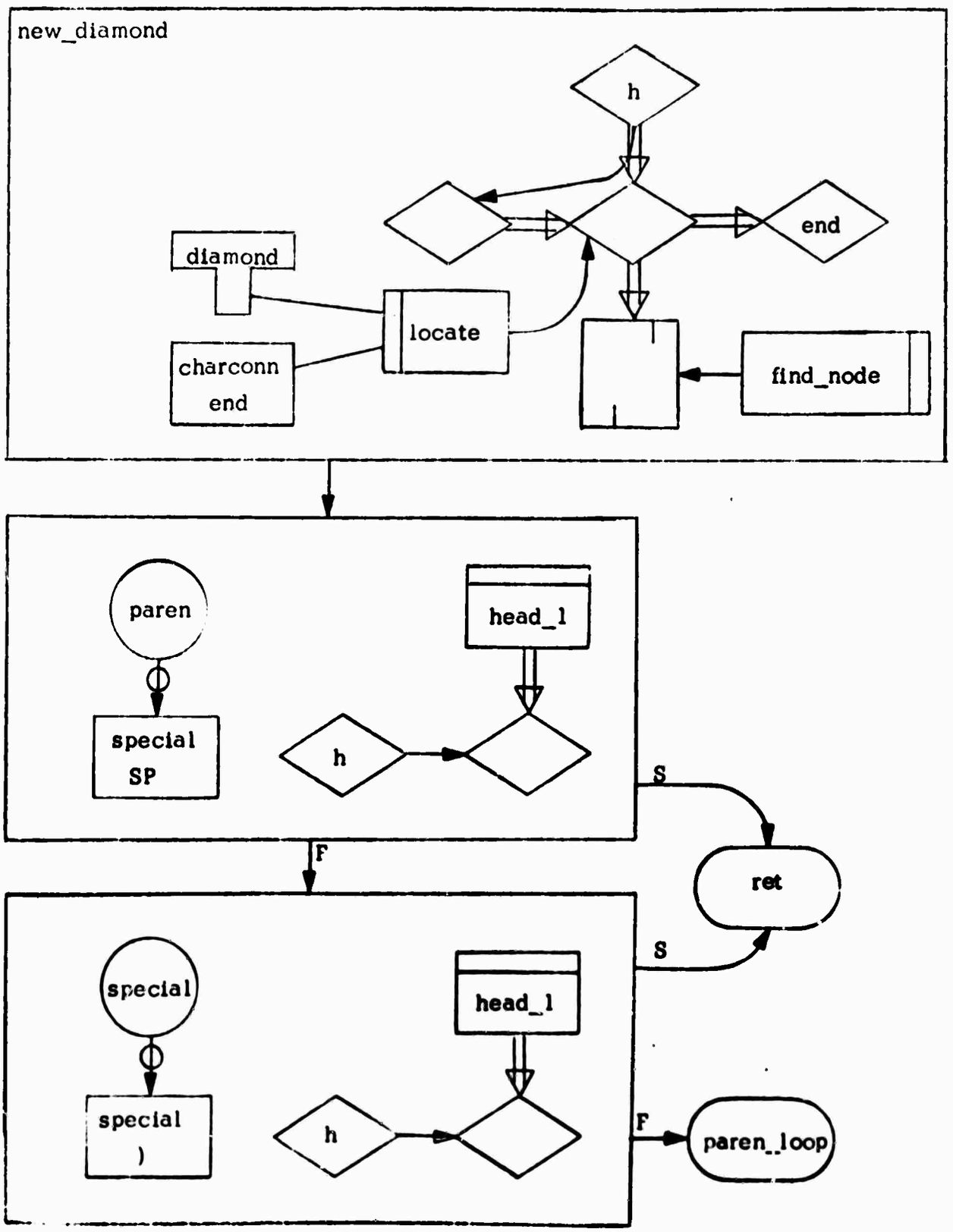

find_noderep_there
next_pname
pname
pname
pname
node
pname
charconn
free
fixed
noderep
variability
head_1
locate
charconn
end
rep
find2
ret

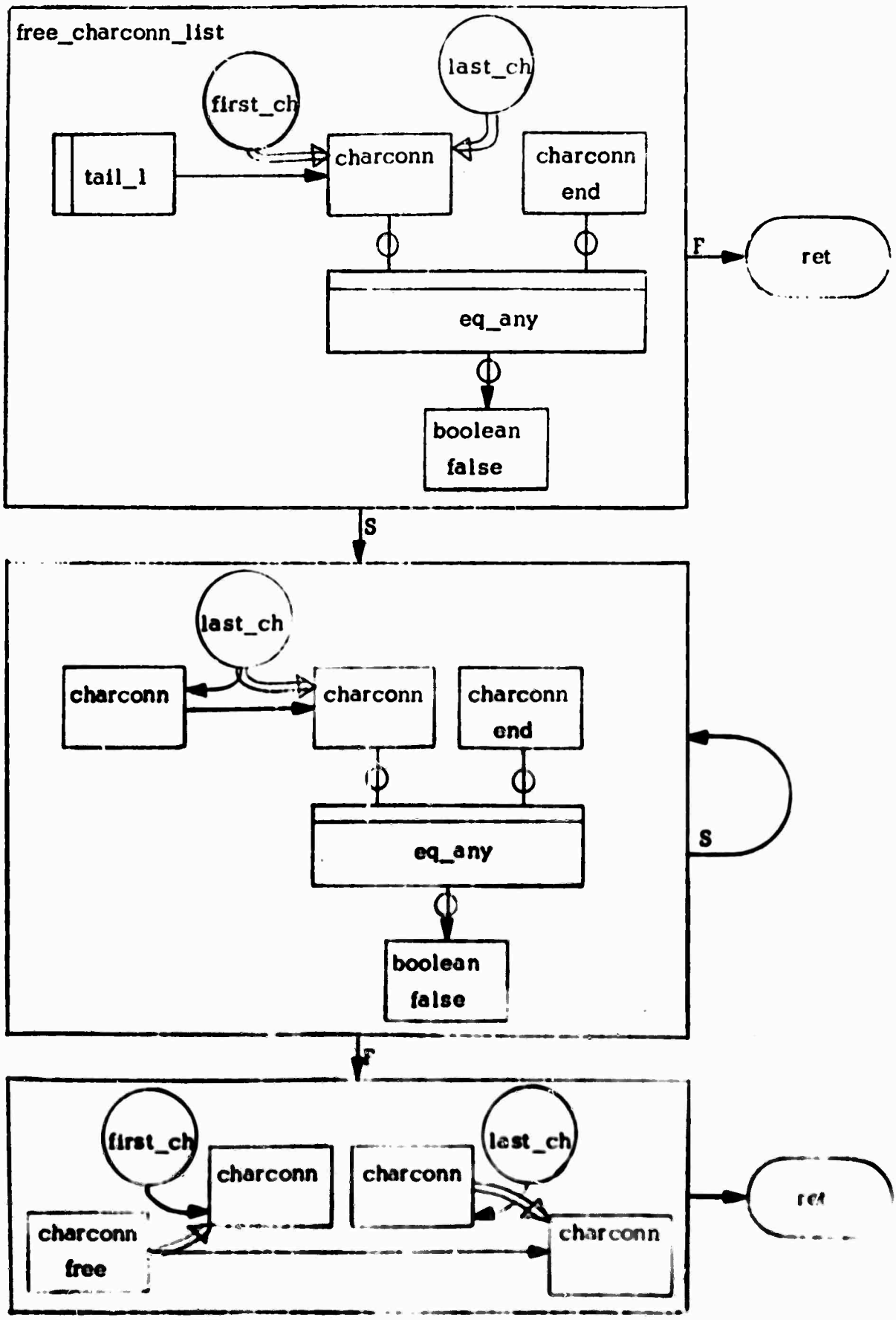

get_link_name
char
charconn
charconn
special
SP
charconn
free
charconn
SF
char
charconn
charconn
special
:
charconn
free
charconn
F
no_colon
S
colon

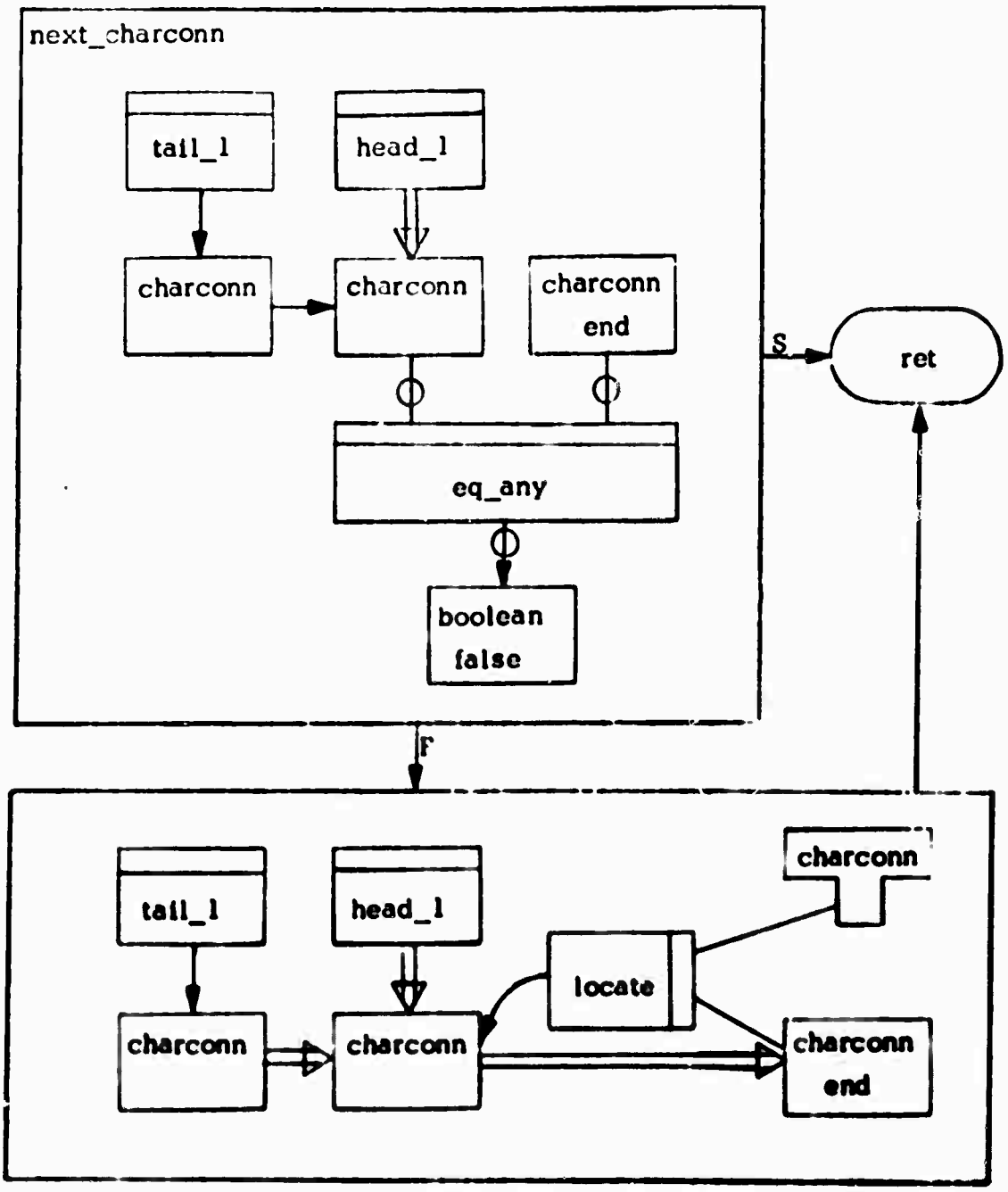

no_colon
head_1
link
link
locate
charconn
get_word
charconn
charconn
free
charconn
ret
colon
head_1
find_node


heads_or_tails
h
end
paren
special
SP
char
charconn
charconn
paren
special
(
charconn
free
charconn
SF
char
charconn
charconn
special
SP
charconn
free
charconn
S
F
S
char
charconn
charconn
special
charconn
free
charconn
paren_loop
F


paren_loop
char
charconn
charconn
special
SP
charconn
free
charconn
SF
char
charconn
charconn
special
)
charconn
free
charconn
h
head_1
F
new_diamond
S
ret


new_diamond
h
end
diamond
locate
charconn
end
find_node
paren
special
SP
head_1
h
S
F
ret
special
special
)
head_1
h
S
F
paren_loop


free_charconn_list
first_ch
last_ch
tail_1
charconn
charconn
end
eq_any
boolean
false
F
ret
S
last_ch
charconn
charconn
charconn
end
eq_any
boolean
false
S
F
first_ch
charconn
charconn
last_ch
charconn
free
charconn
ret


next_charconn
tail_1
head_1
charconn
charconn
charconn
end
eq_any
boolean
false
S
ret
F
tail_1
head_1
charconn
locate
charconn
charconn
charconn
end